Reference 2

(19)日本国特許庁（ＪＰ）　(12)公表特許公報（Ａ）　(11)特許出願公表番号
特表2002－528797
（P2002－528797A）
(43)公表日　平成14年９月３日(2002.9.3)

(51)Int.Cl.'		識別記号	FI		テーマコード'(参考)
G06F	19/00	300	G06F	19/00	300N　5B082
	12/00	546		12/00	546A
		547			547H
	17/60	306		17/60	306
		310			310Z

審査請求　未請求　予備審査請求　未請求(全173頁)　最終頁に続く

(21)出願番号　特願2000－577598(P2000－577598)
(86)(22)出願日　平成11年10月８日(1999.10.8)
(85)翻訳文提出日　平成12年６月16日(2000.6.16)
(86)国際出願番号　ＰＣＴ／ＵＳ99／23426
(87)国際公開番号　ＷＯ００／２３９２５
(87)国際公開日　平成12年４月27日(2000.4.27)
(31)優先権主張番号　０９／１７３，８５８
(32)優先日　平成10年10月16日(1998.10.16)
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　０９／１７３，８４７
(32)優先日　平成10年10月16日(1998.10.16)
(33)優先権主張国　米国（ＵＳ）

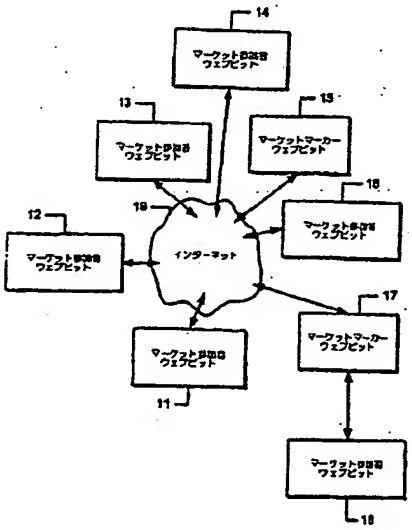
Equivalent to Ref 2

(71)出願人　コマース　ワン　インコーポレイテッド
アメリカ合衆国　カリフォルニア州　94040　マウンテン　ヴィュー　ウェスト　エル　カミノ　リール　2440　セヴンス　フロアー
(72)発明者　メルツァー　バート　アラン
アメリカ合衆国　カリフォルニア州　95003　アプトス　サンタ　マルガリータ　530
(74)代理人　弁理士　中村　稔　（外９名）

最終頁に続く

(54)【発明の名称】　トレーディングパートナネットワークにおける商業のためのドキュメント及びそのドキュメントを基にしたインターフェースの定義

(57)【要約】
機械で読取り可能なドキュメントはビジネスを顧客、供給者、及びトレーディングパートナに結び付ける。ＸＭＬベースのドキュメントのような自己定義電子ドキュメントはパートナ間で容易に理解可能である。ビジネスインターフェース定義と呼ばれるこれらの電子ビジネスドキュメントの定義はインターネット、又はネットワークのメンバに通信する他の方法で送られる。ビジネスインターフェース定義は会社が提供するサービス及びそのようなサービスと通信する時に使用されるドキュメントを潜在的なトレーディングパートナに知らせる。したがって、典型的なビジネスインターフェース定義は購入注文を提示することにより顧客に注文をさせ、供給者は在庫状態レポートをダウンロードすることにより入手可能性をチェックする。また、入力及び出力ドキュメントの合成は共通のビジネスライブラリの翻訳情報と結合して、ペーパベースのビジネスが動作する方法に非常に近似の方法でトランザクションをプログラムする。



【特許請求の範囲】

【請求項１】　トランザクション（ｔｒａｎｓａｃｔｉｏｎ）に関連する処理を実行する複数のノード（ｎｏｄｅｓ）を含むネットワークにおけるノード間のトランザクションのためのインターフェース（ｉｎｔｅｒｆａｃｅ）であって、

入力ドキュメント（ｄｏｃｕｍｅｎｔ）の定義及び出力ドキュメントの定義を供給する翻訳情報を含む、ネットーワークにおける少なくとも一つのノードによってアクセス可能なメモリに記録されたトランザクション処理へのインターフェースの機械可読仕様（ｍａｃｈｉｎｅ　ｒｅａｄａｂｌｅ　ｓｐｅｃｉｆｉｃａｔｉｏｎ）であり、前記入力及び出力ドキュメントの定義は、記録装置の組の個別の記述と、及び前記記録装置の組に対する論理構造を具備する前記仕様を具備することを特徴とするインターフェース。

【請求項２】　前記翻訳情報は、前記入力及び出力ドキュメントの定義における少なくとも一つの論理構造に対するデータ・タイプ（ｄａｔａ　ｔｙｐｅ）仕様を含むことを特徴とする請求項１に記載のインターフェース。

【請求項３】　前記翻訳情報は、前記入力及び出力ドキュメントの定義における特定の論理構造に対する予め決められた組の記録装置を、リスト（ｌｉｓｔ）の個別のエントリ（ｅｎｔｒｉｅｓ）にマッピング（ｍａｐｐｉｎｇ）する少なくとも一つのデータ構造を含むことを特徴とする請求項１に記載のインターフェース。

【請求項４】　論理構造のライブラリと、及び論理構造に対する翻訳情報を記録する、ネットワークにおける少なくとも一つのノードによってアクセス可能であるメモリにおけるリポジトリを含むことを特徴とする請求項１に記載のインターフェース。

【請求項５】　前記機械可読仕様は、特定のトランザクションの識別子を記録するための論理構造と、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項１に記載のインターフェース。

【請求項６】　前記機械可読仕様は、インターフェースの識別子を記録するための、及び前記インターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項１に記載のインターフェース。

【請求項７】　前記機械可読仕様は、特定のトランザクションの仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項６に記載のインターフェース。

【請求項８】　前記記録装置はパースされた（ｐａｒｓｅｄ）データを具備することを特徴とする請求項１に記載のインターフェース。

【請求項９】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタ（ｔｅｘｔ　ｃｈａｒａｃｔｅｒ）をコード化するキャラクタ・データ（ｃｈａｒａｃｔｅｒ　ｄａｔａ）と；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識別するマーク付けデータとを具備することを特徴とする請求項８に記載のインターフェース。

【請求項１０】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項９に記載のインターフェース。

【請求項１１】　前記入力及び出力ドキュメントの少なくとも一つの特定の論理構造によって識別される前記記録装置の組の少なくとも一つに対する前記翻訳情報は、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項８に記載のインターフェース。

【請求項１２】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項８に記載のインターフェース。

【請求項１３】　複数のトランザクションにおける使用のためのドキュメント・タイプのネットワークにおいて、少なくとも一つのノードによってアクセス可能であるメモリに記録されたレポジトリを含み、及び前記入力及び出力ドキュメントの一つの定義は、前記レポジトリにおけるドキュメント・タイプへの参照を含むことを特徴とする請求項１に記載のインターフェース。

【請求項１４】　ドキュメント・タイプの前記レポジトリは、ネットワークにおける参加者処理（ｐａｒｔｉｃｉｐａｎｔ　ｐｒｏｃｅｓｓｅｓ）を識別するためのドキュメント・タイプを含むことを特徴とする請求項１３に記載の方法。

【請求項１５】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項１に記載のインターフェース。

【請求項１６】　翻訳情報を含む前記機械可読データ構造は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義に従って編成されたドキュメントを具備することを特徴とする請求項１に記載のインターフェース。

【請求項１７】　ネットワーク・インターフェースと、及びトランザクション処理アーキテクチャに従って、トランザクションのトランザクション処理を実行するデータ処理リソースを含むシステム上で実行されるトランザクションに対する参加者インターフェース（ｐａｒｔｉｃｉｐａｎｔ　ｉｎｔｅｒｆａｃｅ）を設定するための装置であって：

前記システムによって実行され、前記システムによってアクセス可能な媒体に記録され、特定のトランザクションにおける参加者に対する参加者インターフェースの定義を構築するためのツールを供給する命令のプログラムであって、前記参加者インターフェースの定義は、前記参加者に対する入力ドキュメントの定義と、及び前記参加者に対する出力ドキュメントの定義とを含み、前記入力及び出力ドキュメントの定義は、記録装置の組の個別の機械可読記述と、及び前記記録装置の組に対する論理構造とを具備する前記プログラムと；及び

前記システムによって実行可能であり、前記システムによってアクセス可能な媒体に記録され、前記記録装置の組に対応するデータ構造と、及び前記トランザ

クション処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造とをコンパイルするために、前記入力ドキュメントを、前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令をコンパイルするために、及び前記トランザクション処理の出力を、前記記録装置の組と及び前記出力ドキュメントの論理構造に翻訳するために、前記システムによって実行可能な命令をコンパイルするために、前記入力及び出力ドキュメントの定義に応答するプログラムとを具備することを特徴とする装置。

【請求項１８】　参加者インターフェースの定義を構築するための前記ツールは、レポジトリからの前記定義のエレメント（ｅｌｅｍｅｎｔ）にアクセスするために前記システムによって実行可能である命令を含み、前記レポジトリは、論理構造のライブラリと、及びインターフェース記述を構築するために使用される論理構造に対する翻訳情報とを記録することを特徴とする請求項１７に記載の装置。

【請求項１９】　前記レポジトリは、論理構造を具備するドキュメントの定義を記録することを特徴とする請求項１８に記載の装置。

【請求項２０】　参加者インターフェースの定義を構築するための前記ツールは、

相補トランザクションに対する他の参加者インターフェースの定義にアクセスするためであり、前記アクセスされた定義は、前記相補トランザクションに対する入力ドキュメントの定義と、及び前記相補トランザクションに対する出力ドキュメントの定義を含む前記アクセスのために；及び

構築されている前記インターフェースの入力ドキュメントの定義のように、前記相補トランザクションの前記出力ドキュメントの定義を含むことによって、前記参加者インターフェースの定義を設定するために、前記システムによって実行可能な命令を含むことを特徴とする請求項１７に記載の装置。

【請求項２１】　参加者インターフェースの定義を構築するための前記ツールは、

構築されているインターフェースの前記出力ドキュメントの定義のように、前記相補トランザクションの入力ドキュメントの定義を含むために、

前記システムによって実行可能である命令を含むことを特徴とする請求項２０に記載の装置。

【請求項２２】　参加者インターフェースの定義を構築するための前記ツールは、特定のトランザクションの識別子を記録するための論理構造を含む参加者インターフェース・ドキュメントの定義に従ったドキュメントと、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つとを構築するために、前記システムによって実行可能である命令を含むことを特徴とする請求項１７に記載の装置。

【請求項２３】　参加者インターフェースの定義を構築するための前記ツールは、前記参加者インターフェースの識別子を記録するため、及び前記参加者インターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含む参加者インターフェース・ドキュメントの定義に従ったドキュメントを構築するために、前記システムによって実行可能な命令を含むことを特徴とする請求項１７に記載の装置。

【請求項２４】　参加者インターフェース・ドキュメントの定義に従った前記ドキュメントは、特定のトランザクションの機械可読仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項２３に記載の装置。

【請求項２５】　前記記録装置は、パースされたデータを具備することを特徴とする請求項１７に記載の装置。

【請求項２６】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタ（ｔｅｘｔ　ｃｈａｒａｃｔｅｒ）をコード化するキャラクタ・データ（ｃｈａｒａｃｔｅｒ　ｄａｔａ）と；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識

別するマーク付けデータとを具備することを特徴とする請求項２５に記載の装置。

【請求項２７】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項２６に記載の装置。

【請求項２８】　前記仕様は、前記入力及び出力ドキュメントのうち少なくとも一つの前記論理構造によって識別される前記記録装置の組のうち少なくとも一つに対する翻訳情報を含み、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項２５に記載の装置。

【請求項２９】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項２５に記載の装置。

【請求項３０】　前記記録装置の組に対応するデータ構造及び前記入力及び出力ドキュメントの論理構造は、可変トランザクション処理アーキテクチャに従った変数及び方法を含むプログラミング・オブジェクトを含むことを特徴とする請求項１７に記載の装置。

【請求項３１】　前記トランザクション処理の前記可変トランザクション処理アーキテクチャは、インターフェース記述言語に従った処理を具備することを特徴とする請求項１７に記載の装置。

【請求項３２】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項１７に記載の装置。

【請求項３３】　前記入力及び出力ドキュメントの一つの前記定義は、レポジトリにおけるドキュメント・タイプへの参照を含むことを特徴とする請求項１９に記載の装置。

【請求項３４】　前記レポジトリは、トランザクションの製品サブジェクトの測定を特定する翻訳情報を含むことを特徴とする請求項１８に記載の装置。

【請求項３５】　前記レポジトリは、トランザクションの製品サブジェクトのコスト（ｃｏｓｔ）を特定する翻訳情報を含むことを特徴とする請求項１８に記載の装置。

【請求項３６】　前記レポジトリは、トランザクションの製品サブジェクトの特性を特定する翻訳情報を含むことを特徴とする請求項１８に記載の装置。

【請求項３７】　前記レポジトリは、トランザクションの金融ターム（ｆｉｎａｎｃｉａｌ　ｔｅｒｍｓ）を特定する翻訳情報を含むことを特徴とする請求項１８に記載の装置。

【請求項３８】　前記レポジトリは、トランザクションの製品サブジェクトに対する出荷（ｓｈｉｐｍｅｎｔ）のタームを特定する翻訳情報を含むことを特徴とする請求項１８に記載の装置。

【請求項３９】　システム上で実行されるトランザクションに対する参加者インターフェースを設定するための装置であって：

命令のデータ及びプログラムを記録するメモリと；

命令の前記プログラムを実行するメモリに接続されたデータ・プロセッサであって；前記命令のプログラムは、

特定のトランザクションにおける参加者のための参加者インターフェースの定義を構築するためのツールであって、参加者インターフェースの前記定義は、前記参加者に対する入力ドキュメントの定義及び前記参加者に対する出力ドキュメントの定義を含み、前記入力及び出力ドキュメントの定義は、記録装置の組の個別の機械可読記述と、及び前記記録装置の組に対する論理構造とを具備する前記ツールと；及び

前記入力及び出力ドキュメントの定義に応答し、前記記録装置の組と及び前記トランザクション処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造とに対応したデータ構造をコンパイルするため、前記入力ドキュメントを前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令をコンパイルし、及び前記トランザクション処理の出力を、前記記録装置の組と及び前記出力ドキュメントの論理構造とに翻訳するために、前記システムによって実行可能な命令をコンパイルするためのコンパイラとを含む前記データ・プロセッサとを具備することを特徴とする装置。

【請求項４０】　前記データ・プロセッサによってアクセス可能なメモリに記録されたレポジトリを含み、参加者インターフェースの定義を構築するための

前記ツールは、レポジトリからの前記定義のエレメントにアクセスするために前記システムによって実行可能である命令を含み、前記レポジトリは、論理構造のライブラリと、及びインターフェース記述を構築するために使用される論理構造に対する翻訳情報とを記録することを特徴とする請求項３９に記載の装置。

【請求項４１】　前記レポジトリは、論理構造を具備するドキュメントの定義を記録することを特徴とする請求項４０に記載の装置。

【請求項４２】　参加者インターフェースの定義を構築するための前記ツールは、

相補トランザクションに対する他の参加者インターフェースの定義にアクセスするためであり、前記アクセスされた定義は、前記相補トランザクションに対する入力ドキュメントの定義と、及び前記相補トランザクションに対する出力ドキュメントの定義を含む前記アクセスのために；及び

構築されている前記インターフェースの入力ドキュメントの定義のように、前記相補トランザクションの前記出力ドキュメントの定義を含むことによって、前記参加者インターフェースの定義を設定するために、前記システムによって実行可能な命令を含むことを特徴とする請求項３９に記載の装置。

【請求項４３】　参加者インターフェースの定義を構築するための前記ツールは、

構築されているインターフェースの前記出力ドキュメントの定義のように、前記相補トランザクションの入力ドキュメントの定義を含むために、前記システムによって実行可能である命令を含むことを特徴とする請求項４２に記載の装置。

【請求項４４】　参加者インターフェースの定義を構築するための前記ツールは、特定のトランザクションの識別子を記録するための論理構造を含む参加者インターフェース・ドキュメントの定義に従ったドキュメントと、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つとを構築するために、前記システムによって実行可能である命令を含むことを特徴とする請求項３９に記載の装置。

【請求項４５】　参加者インターフェースの定義を構築するための前記ツールは、前記参加者インターフェースの識別子を記録するため、及び前記参加者イ

ンターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含む参加者インターフェース・ドキュメントの定義に従ったドキュメントを構築するために、前記システムによって実行可能な命令を含むことを特徴とする請求項３９に記載の装置。

【請求項４６】　参加者インターフェース・ドキュメントの定義に従った前記ドキュメントは、特定のトランザクションの機械可読仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項４５に記載の装置。

【請求項４７】　前記記録装置は、パースされたデータを具備することを特徴とする請求項２５に記載の装置。

【請求項４８】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタをコード化するキャラクタ・データと；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識別するマーク付けデータとを具備することを特徴とする請求項４７に記載の装置。

【請求項４９】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項４８に記載の装置。

【請求項５０】　前記仕様は、前記入力及び出力ドキュメントのうち少なくとも一つの前記論理構造によって識別される前記記録装置の組のうち少なくとも一つに対する翻訳情報を含み、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項４７に記載の装置。

【請求項５１】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項４７に記載の装置。

【請求項５２】　前記記録装置の組に対応するデータ構造及び前記入力及び出力ドキュメントの論理構造は、可変トランザクション処理アーキテクチャに従った変数及び方法を含むプログラミング・オブジェクトを含むことを特徴とする請求項３９に記載の装置。

【請求項５３】　前記トランザクション処理の前記可変トランザクション処理アーキテクチャは、インターフェース記述言語に従った処理を具備することを特徴とする請求項３９に記載の装置。

【請求項５４】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項３９に記載の装置。

【請求項５５】　前記入力及び出力ドキュメントの一つの前記定義は、レポジトリにおけるドキュメント・タイプへの参照を含むことを特徴とする請求項４１に記載の装置。

【請求項５６】　前記レポジトリは、トランザクションの製品サブジェクトの測定を特定する翻訳情報を含むことを特徴とする請求項４０に記載の装置。

【請求項５７】　前記レポジトリは、トランザクションの製品サブジェクトのコストを特定する翻訳情報を含むことを特徴とする請求項４０に記載の装置。

【請求項５８】　前記レポジトリは、トランザクションの製品サブジェクトの特性を特定する翻訳情報を含むことを特徴とする請求項４０に記載の装置。

【請求項５９】　前記レポジトリは、トランザクションの金融タームを特定する翻訳情報を含むことを特徴とする請求項４０に記載の装置。

【請求項６０】　前記レポジトリは、トランザクションの製品サブジェクトに対する出荷のタームを特定する翻訳情報を含むことを特徴とする請求項４０に記載の装置。

【請求項６１】　ネットワークにおいて商業トランザクションをプログラミングするための方法であって：

前記トランザクションにおける処理を実行するためのリソースを含むネットワークにおけるノードに対する入力ドキュメントの機械可読定義と、及び前記ノードに対する出力ドキュメントの機械可読定義を定義する段階であって、前記入力

及び出力ドキュメントの定義は、記録装置の組の個別の記述と及び前記記録装置の組に対する論理構造とを具備する前記段階と；及び

前記論理構造に対する翻訳情報を前記ノードに供給する段階とを具備することを特徴とする方法。

【請求項６２】　前記翻訳情報は、前記入力及び出力ドキュメントの定義における少なくとも一つの論理構造に対するデータ・タイプ仕様を含むことを特徴とする請求項６１に記載の方法。

【請求項６３】　前記翻訳情報は、前記入力及び出力ドキュメントの定義における特定の論理構造に対する予め決められた組の記録装置を、リストの個別のエントリにマッピングする少なくとも一つのデータ構造を含むことを特徴とする請求項前記翻訳情報は、前記入力及び出力ドキュメントの定義における少なくとも一つの論理構造に対するデータ・タイプの仕様を含むことを特徴とする請求項６１に記載の方法。

【請求項６４】　翻訳情報を供給する前記段階は、論理構造のライブラリと、及び論理構造に対する翻訳情報を記録する、ネットワークにおける少なくとも一つのノードによってアクセス可能であるメモリにおけるリポジトリを供給する段階を含むことを特徴とする請求項６１に記載の方法。

【請求項６５】　特定のトランザクションの識別子を記録するための論理構造と、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むインターフェース・ドキュメントの定義に従ったドキュメントを含むインターフェースの機械可読仕様を定義する段階を含むことを特徴とする請求項６１に記載の方法。

【請求項６６】　前記記録装置はパースされたデータを具備することを特徴とする請求項６１に記載の方法。

【請求項６７】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタをコード化するキャラクタ・データと；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識別するマーク付けデータとを具備することを特徴とする請求項６６に記載の方法。

【請求項６８】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項６７に記載の方法。

【請求項６９】　前記入力及び出力ドキュメントの少なくとも一つの特定の論理構造によって識別される前記記録装置の組の少なくとも一つに対する前記翻訳情報は、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項６７に記載の方法。

【請求項７０】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項６６に記載の方法。

【請求項７１】　前記組の記録装置の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項６７に記載の方法。

【請求項６９】　前記入力及び出力ドキュメントの少なくとも一つの特定の論理構造によって識別される前記組の記録装置の少なくとも一つに対する前記翻訳情報は、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項６７に記載の方法。

【請求項７０】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項６６に記載の方法。

【請求項７１】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項６１に記載の方法。

【請求項７２】　前記入力ドキュメントの定義において、論理構造に応答してイベント信号（ｅｖｅｎｔ　ｓｉｇｎａｌ）を生成するためのパーサ（ｐａｒｓｅｒ）を供給する段階と；及び

前記処理を実行するために、前記イベント信号に応答するイベント・リスナ・プログラム（ｅｖｅｎｔ　ｌｉｓｔｅｎｅｒ　ｐｒｏｇｒａｍｓ）を供給する段

階とを含むことを特徴とする請求項６１に記載の方法。

【請求項７３】　トランザクションに関連する処理を実行する複数のノードを含むネットワークにおけるノード間のトランザクションを実行するための方法であって：

トランザクションに対するインターフェースの機械可読仕様を記録する段階であって、前記仕様は入力ドキュメントの定義及び出力ドキュメントの定義を含み、前記入力及び出力ドキュメントの定義は、記録装置の組の個別の記述と及び前記記録装置の組に対する論理構造とを具備する前記段階と；

通信ネットワークを通してドキュメントを具備するデータを受信する段階と；

入力ドキュメントを識別するための前記仕様に従って前記ドキュメントをパースする段階と；

機械可読形式での前記入力ドキュメントの少なくとも一部分を、出力を生成するトランザクション処理に供給する段階と；

前記仕様に基づいて、出力ドキュメントの前記定義に従った前記出力を具備する出力ドキュメントを形成する段階と；及び

前記通信ネットワーク上で前記出力ドキュメントを転送する段階とを具備することを特徴とする方法。

【請求項７４】　前記トランザクションに対するネットワークにおける他のノードに供給される相補インターフェースの仕様にアクセスする段階であって、前記アクセスされた仕様は、前記相補インターフェースに対する入力ドキュメントの定義と、及び前記相補インターフェースに対する出力ドキュメントの定義とを含む前記段階と；及び

前記記録された仕様におけるインターフェースの前記入力ドキュメントの前記定義における前記相補インターフェースの前記出力ドキュメントの前記定義の少なくとも一部を含むことによって、前記インターフェースの前記記録された仕様を設定する段階とを含むことを特徴とする請求項７３に記載の方法。

【請求項７５】　ネットワークにおいて前記相補インターフェースを見つける段階を含むことを特徴とする請求項７４に記載の方法。

【請求項７６】　前記記録された仕様を設定する前記段階は、レポジトリか

らの機械可読仕様のエレメントにアクセスする段階であって、前記レポジトリは、論理構造のライブラリと、論理構造に対する模式図マップ（ｓｃｈｅｍａｔｉｃ　ｍａｐ）と、及びインターフェース記述を構築するために使用される論理構造を具備するドキュメントの定義とを記録する前記段階を含むことを特徴とする請求項７４に記載の方法。

【請求項７７】　前記記録された仕様におけるインターフェースの前記出力ドキュメントの前記定義における前記相補インターフェースの前記入力ドキュメントの前記定義の少なくとも一部を含むことによって、前記インターフェースの前記記録された仕様を設定する段階とを含むことを特徴とする請求項７４に記載の方法。

【請求項７８】　通信ネットワークを通した前記仕様へのアクセスを、ネットーワークにおける他のノードに供給する段階を含むことを特徴とする請求項７３に記載の方法。

【請求項７９】　前記インターフェースの仕様をネットワークにおける他のノードに送信する段階を含み、前記ネットワークにおいては、前記仕様へのアクセスはネットワークにおける他のノードに対して供給される前記段階を含むことを特徴とする請求項７３に記載の方法。

【請求項８０】　前記機械可読仕様は、特定のトランザクションの識別子を記録するための論理構造と、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項７３に記載の方法。

【請求項８１】　前記機械可読仕様は、インターフェースの識別子を記録するための、及び前記インターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項７３に記載の方法。

【請求項８２】　前記機械可読仕様は、特定のトランザクションの仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザク

ションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項８１に記載の方法。

【請求項８３】　前記記録装置はパースされた（ｐａｒｓｅｄ）データを具備することを特徴とする請求項７３に記載の方法。

【請求項８４】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタをコード化するキャラクタ・データと；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識別するマーク付けデータとを具備することを特徴とする請求項８３に記載の方法。

【請求項８５】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項８４に記載の方法。

【請求項８６】　前記仕様は、前記入力及び出力ドキュメントのうち少なくとも一つの論理構造によって識別される前記記録装置の組のうち少なくとも一つに対する前記翻訳情報を含み、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項８３に記載の方法。

【請求項８７】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項８３に記載の方法。

【請求項８８】　前記トランザクション処理は、可変トランザクション処理アーキテクチャを有し、及び前記入力ドキュメントの少なくとも一部分を、前記トランザクション処理の前記可変トランザクション処理アーキテクチャに従って読み取り可能な形式に翻訳する段階を含むことを特徴とする請求項７３に記載の方法。

【請求項８９】　前記翻訳段階は、前記トランザクション処理の前記可変トランザクション処理アーキテクチャに従った変数及び方法を含むプログラミング・オブジェクトを生成する段階を含むことを特徴とする請求項８８に記載の方法

。

【請求項９０】　前記トランザクション処理の前記可変トランザクション処理アーキテクチャは、インターフェース記述言語に従った処理を具備することを特徴とする請求項８８に記載の方法。

【請求項９１】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項７３に記載の方法。

【請求項９２】　複数のトランザクションにおける使用のためのドキュメント・タイプのレポジトリを供給する段階を含み、及び前記入力及び出力ドキュメントの一つの前記定義は、前記レポジトリにおけるドキュメント・タイプへの参照を含むことを特徴とする請求項７３に記載の方法。

【請求項９３】　ドキュメント・タイプの前記レポジトリは、ネットワークにおける参加者処理を識別するためのドキュメント・タイプを含むことを特徴とする請求項９２に記載の方法。

【請求項９４】　論理構造に対する翻訳情報のレポジトリを供給する段階を含み、トランザクションのパラメータを識別する翻訳情報を含むことを特徴とする請求項９２に記載の方法。

【請求項９５】　前記トランザクション処理は、可変トランザクション処理アーキテクチャを有し、及び：

相補インターフェースの仕様にアクセスする段階であって、前記アクセスされた仕様は前記相補インターフェースに対する入力ドキュメントの定義と及び前記相補インターフェースに対する出力ドキュメントの定義とを含む前記段階と；

記録された仕様におけるインターフェースの前記入力ドキュメントの定義のように、前記相補インターフェースの前記出力ドキュメントの定義を含むことによって、インターフェースの記録された仕様を設定する段階と；及び

前記入力及び出力ドキュメントの定義に応答して、記録装置の組に対応したデータ構造及び前記トランザクション処理の前記トランザクション処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造と、前記入力ドキュメントを前記対応するデータ構造に翻訳するために、前記システムによって実行可能

な命令と、及び前記トランザクション処理の出力を前記出力ドキュメントの定義に従った形式に翻訳するために、前記システムによって実行可能な命令とをコンパイルする段階とを含むことを特徴とする請求項７３に記載の方法。

【請求項９６】　記録された仕様におけるインターフェースの前記出力ドキュメントの定義のように、前記相補インターフェースの前記入力ドキュメントの定義を含むことによって、インターフェースの記録された仕様を設定する段階を含むことを特徴とする請求項９５に記載の方法。

【請求項９７】　前記トランザクション処理は、可変トランザクション処理アーキテクチャを有し、及び前記記録された仕様を設定する段階は、レポジトリからの前記機械可読仕様のエレメントにアクセスする段階を含み、前記レポジトリは論理構造のライブラリと、論理構造に対する模式図マップと、及びインターフェース記述を構築するために使用される論理構造を具備するドキュメントの定義とを記録し、及び；

前記入力及び出力ドキュメントの定義に応答して、記録装置の組に対応したデータ構造及び前記トランザクション処理の前記トランザクション処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造と、前記入力ドキュメントを前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令と、及び前記トランザクション処理の出力を前記出力ドキュメントの定義に従った形式に翻訳するために、前記システムによって実行可能な命令とをコンパイルする段階とを含むことを特徴とする請求項７３に記載の方法。

【請求項９８】　トランザクションに関連する処理を実行する複数のノードを含むネットワークにおけるノード間のトランザクションを管理するための装置であって：

ネットワーク・インターフェースと；

命令のデータ及びプログラムを記録するメモリであって、トランザクションに対するインターフェースの機械可読仕様を含み、前記仕様は入力ドキュメントの定義と及び出力ドキュメントの定義とを含み、前記入力及び出力ドキュメントの定義は、記録装置の組の個別の記述と及び前記記録装置の組に対する論理構造とを具備する前記メモリと；

前記メモリと及び命令のプログラムを実行する前記ネットワーク・インターフェースとに接続されたデータ・プロセッサであって；前記命令のプログラムは、

ネットワーク・インターフェースを通してドキュメントを具備するデータを受信するための論理と；

入力ドキュメントを識別するための前記仕様に従って、前記ドキュメントをパースするための論理と；

機械可読形式での前記入力ドキュメントの少なくとも一部分を、出力を生成するトランザクション処理に供給するための論理と；

前記仕様に基づいて、出力ドキュメントの定義に従って、前記出力を具備する出力ドキュメントを形成するための論理と；及び

ネットワーク・インターフェース上で前記出力ドキュメントを転送するための論理とを含む前記データ・プロセッサとを具備することを特徴とする装置。

【請求項９９】　相補インターフェースの仕様にアクセスするための論理であって、前記アクセスされた仕様は前記相補インターフェースに対する入力ドキュメントの定義と及び前記相補インターフェースに対する出力ドキュメントの定義とを含む前記論理と；及び

記録された仕様における前記インターフェースの前記入力ドキュメントの定義のように、前記相補インターフェースの前記出力ドキュメントの定義を含むことによって、前記インターフェースの前記記録された仕様を設定するための論理とを含むことを特徴とする請求項９８に記載の装置。

【請求項１００】　前記記録された仕様を設定するための前記論理は、レポジトリからの前記機械可読仕様のエレメントにアクセスするための論理を含み、前記レポジトリは論理構造のライブラリと、論理構造に対する模式図マップと、及びインターフェース記述を構築するために使用される論理構造を具備するドキュメントの定義とを記録することを特徴とする請求項９９に記載の装置。

【請求項１０１】　記録された仕様における前記インターフェースの前記出力ドキュメントの定義のように、前記相補インターフェースの前記入力ドキュメントの定義を含むことによって、前記インターフェースの前記記録された仕様を設定するための論理を含むことを特徴とする請求項９９に記載の装置。

【請求項１０２】　通信ネットワークを通した前記仕様へのアクセスを、ネットーワークにおける他のノードに供給する論理を含むことを特徴とする請求項９８に記載の装置。

【請求項１０３】　前記インターフェースの仕様をネットワークにおける他のノードに送信するための論理を含み、前記ネットワークにおいては、前記仕様へのアクセスはネットワークにおける他のノードに対して供給される前記論理を含むことを特徴とする請求項９８に記載の装置。

【請求項１０４】　前記機械可読仕様は、特定のトランザクションの識別子を記録するための論理構造と、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項９８に記載の装置。

【請求項１０５】　前記機械可読仕様は、インターフェースの識別子を記録するための、及び前記インターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項９８に記載の装置。

【請求項１０６】　前記機械可読仕様は、特定のトランザクションの仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項１０５に記載の装置。

【請求項１０７】　前記記録装置はパースされたデータを具備することを特徴とする請求項９８に記載の装置。

【請求項１０８】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタをコード化するキャラクタ・データと；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識

別するマーク付けデータとを具備することを特徴とする請求項１０７に記載の装置。

【請求項１０９】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項１０８に記載の装置。

【請求項１１０】　前記仕様は、前記入力及び出力ドキュメントのうち少なくとも一つの論理構造によって識別される前記記録装置の組のうち少なくとも一つに対する前記翻訳情報を含み、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項１０７に記載の装置。

【請求項１１１】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項１０７に記載の装置。

【請求項１１２】　前記トランザクション処理は、可変トランザクション処理アーキテクチャを有し、及び前記入力ドキュメントの少なくとも一部分を、前記トランザクション処理の前記可変トランザクション処理アーキテクチャに従って読み取り可能な形式に翻訳する論理を含むことを特徴とする請求項９８に記載の装置。

【請求項１１３】　前記翻訳するための論理は、前記トランザクション処理の前記可変トランザクション処理アーキテクチャに従った変数及び方法を含むプログラミング・オブジェクトを生成する論理を含むことを特徴とする請求項１１２に記載の装置。

【請求項１１４】　前記トランザクション処理の前記可変トランザクション処理アーキテクチャは、インターフェース記述言語に従った処理を含むことを特徴とする請求項１１２に記載の装置。

【請求項１１５】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項９８に記載の装置。

【請求項１１６】　複数のトランザクションにおける使用のためのドキュメント・タイプのレポジトリを記録するプロセッサによってアクセス可能なメモリを含み、及び前記入力及び出力ドキュメントの一つの前記定義は、前記レポジト

リにおけるドキュメント・タイプへの参照を含むことを特徴とする請求項９８に記載の装置。

【請求項１１７】　ドキュメント・タイプの前記レポジトリは、ネットワークにおける参加者処理を識別するためのドキュメント・タイプを含むことを特徴とする請求項１１６に記載の装置。

【請求項１１８】　論理構造に対する翻訳情報のレポジトリを供給する段階を含み、トランザクションのパラメータを識別する翻訳情報を含むことを特徴とする請求項１１６に記載の装置。

【請求項１１９】　前記トランザクション処理は、可変トランザクション処理アーキテクチャを有し、及び：

相補インターフェースの仕様にアクセスするための論理であって、前記アクセスされた仕様は前記相補インターフェースに対する入力ドキュメントの定義と及び前記相補インターフェースに対する出力ドキュメントの定義とを含む前記論理と；

記録された仕様におけるインターフェースの前記入力ドキュメントの定義のように、前記相補インターフェースの前記出力ドキュメントの定義を含むことによって、インターフェースの記録された仕様を設定する論理と；及び

前記入力及び出力ドキュメントの定義に応答して、記録装置の組に対応したデータ構造及び前記トランザクション処理の前記トランザクション処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造とをコンパイルするためと、前記入力ドキュメントを前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令をコンパイルするためと、及び前記トランザクション処理の出力を前記出力ドキュメントの定義に従った形式に翻訳するために、前記システムによって実行可能な命令をコンパイルするための論理を含むことを特徴とする請求項９８に記載の装置。

【請求項１２０】　記録された仕様におけるインターフェースの前記出力ドキュメントの定義のように、前記相補インターフェースの前記入力ドキュメントの定義を含むことによって、インターフェースの記録された仕様を設定するための論理を含むことを特徴とする請求項１１９に記載の装置。

【請求項１２１】　前記トランザクション処理は、可変トランザクション処理アーキテクチャを有し、及びレポジトリからの前記機械可読仕様のエレメントにアクセスすることによって、前記記録された仕様を設定するための論理を含み、前記レポジトリは論理構造のライブラリと、論理構造に対する模式図マップと、及びインターフェース記述を構築するために使用される論理構造を具備するドキュメントの定義とを記録し、及び；

前記入力及び出力ドキュメントの定義に応答して、記録装置の組に対応したデータ構造及び前記トランザクション処理の前記トランザクション処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造とをコンパイルするためと、前記入力ドキュメントを前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令をコンパイルするためと、及び前記トランザクション処理の出力を前記出力ドキュメントの定義に従った形式に翻訳するために、前記システムによって実行可能な命令をコンパイルする論理を含むことを特徴とする請求項９８に記載の装置。

【請求項１２２】　トランザクションに関連する処理を実行する複数のノードを含むネットワークにおけるノード間のトランザクションを管理するための装置であって：

複数の参加者インターフェースの機械可読仕様を記録する段階であって、前記参加者インターフェースはトランザクションを識別し、前記個別のトランザクションは入力ドキュメントの定義及び出力ドキュメントの定義によって識別され、前記入力及び出力ドキュメントの前記定義は、記録装置の組の個別の記述と及び前記記録装置の組に対する論理構造とを具備する前記段階と；

通信ネットワークを通してドキュメントを具備するデータを受信する段階と；

入力ドキュメントを識別するための前記仕様と、及び前記識別された入力ドキュメントを受信する一つ以上のトランザクションに従って、前記ドキュメントをパースする段階と；

機械可読形式での前記入力ドキュメントの少なくとも一部分を、前記一つ以上の識別されたトランザクションと関連したトランザクション処理に供給する段階とを具備することを特徴とする方法。

【請求項１２３】　論理構造のライブラリと、論理構造に対する模式図マップと、及び参加者インターフェース記述を構築するために使用される論理構造を具備するドキュメントの定義とを記録するレポジトリを供給する段階を含むことを特徴とする請求項１２２に記載の方法。

【請求項１２４】　通信ネットワークを通した前記レポジトリへのアクセスを、ネットーワークにおける他のノードに供給する段階を含むことを特徴とする請求項１２３に記載の方法。

【請求項１２５】　前記機械可読仕様は、特定のトランザクションの識別子を記録するための論理構造と、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つとを含む参加者インターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項１２２に記載の方法。

【請求項１２６】　前記機械可読仕様は、参加者インターフェースの識別子を記録するための、及び前記参加者インターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項１２２に記載の方法。

【請求項１２７】　参加者インターフェース・ドキュメントの定義に従った前記ドキュメントは、特定のトランザクションの仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項１２６に記載の方法。

【請求項１２８】　前記記録装置はパースされたデータを具備することを特徴とする請求項１２２に記載の方法。

【請求項１２９】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタをコード化するキャラクタ・データと；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識別するマーク付けデータとを具備することを特徴とする請求項１２８に記載の方法。

【請求項１３０】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項１０９に記載の方法。

【請求項１３１】　前記仕様は、前記入力及び出力ドキュメントのうち少なくとも一つの論理構造によって識別される前記記録装置の組のうち少なくとも一つに対する前記翻訳情報を含み、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項１３０に記載の方法。

【請求項１３２】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項１３０に記載の方法。

【請求項１３３】　機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給する段階は、処理アーキテクチャに従ってルーティング処理を実行する段階を含み、及び：

前記参加者インターフェースにおける前記入力及び出力ドキュメントの定義に応答して、記録装置の前記組に対応するデータ構造及び前記トランザクション処理の処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造と、及び前記入力ドキュメントを前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令とをコンパイルする段階を含むことを特徴とする請求項１２２に記載の方法。

【請求項１３４】　機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給する段階は、処理アーキテクチャに従ってルーティング処理を実行する段階と、及び前記入力ドキュメントの少なくとも一部分を、前記処理アーキテクチャに従って読み取り可能である形式に翻訳する段階を含むことを特徴とする請求項１２２に記載の方法。

【請求項１３５】　前記翻訳段階は、前記ルーティング処理の前記処理アー

キテクチャに従った変数及び方法を含むプログラミング・オブジェクトを生成する段階を含むことを特徴とする請求項１３４に記載の方法。

【請求項１３６】　機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給する段階は、前記入力ドキュメントの前記部分を前記識別されたトランザクションにルーティングする段階を含むことを特徴とする請求項１２２に記載の方法。

【請求項１３７】　前記ルーティング段階は、通信ネットワーク上の前記入力ドキュメントを、前記識別されたトランザクションの一つを実行するノードに送信する段階を含むことを特徴とする請求項１３６に記載の方法。

【請求項１３８】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項１２２に記載の方法。

【請求項１３９】　参加者インターフェースの仕様は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義に従ってドキュメントの定義を具備することを特徴とする請求項１３８に記載の方法。

【請求項１４０】　前記レポジトリは複数のトランザクションにおける使用のための標準化されたドキュメント・タイプを含み、及び前記入力及び出力ドキュメントの一つの前記定義は、レポジトリにおける標準化されたドキュメント・タイプへの参照を含むことを特徴とする請求項１２２に記載の方法。

【請求項１４１】　前記レポジトリは、ネットワークにおける参加者処理を識別するための標準化されたドキュメント・タイプを含むことを特徴とする請求項１４０に記載の方法。

【請求項１４２】　論理構造に対する翻訳情報のレポジトリを供給する段階を含み、トランザクションのパラメータを識別する翻訳情報を含むことを特徴とする請求項１４０に記載の方法。

【請求項１４３】　前記トランザクション処理は、複数の可変トランザクション処理アーキテクチャの一つを各々有し、及び前記入力ドキュメントの少なくとも一部分を、前記個別のトランザクション処理の前記可変トランザクション処

理アーキテクチャに従って読み取り可能な形式に翻訳する段階と、及び前記翻訳された部分を前記個別のトランザクション処理にルーティングする段階とを含むことを特徴とする請求項１２２に記載の方法。

【請求項１４４】　前記翻訳段階は、前記個別のトランザクション処理の可変トランザクション処理アーキテクチャに従って変数及び方法を含むプログラミング・オブジェクトを生成する段階を含むことを特徴とする請求項１４３に記載の方法。

【請求項１４５】　前記トランザクション処理の前記可変トランザクション処理アーキテクチャは、インターフェース記述言語に従った処理を具備することを特徴とする請求項１４４に記載の方法。

【請求項１４６】　トランザクションに関連する処理を実行する複数のノードを含むネットワークにおけるノード間のトランザクションを管理するための装置であって：

ネットワーク・インターフェースと；

命令のデータ及びプログラムを記録するメモリであって、複数の参加者インターフェースの機械可読仕様を含み、前記参加者インターフェースはトランザクションを識別し、前記個別のトランザクションは、入力ドキュメントの定義及び出力ドキュメントの定義によって識別され、前記入力及び出力ドキュメントの定義は、記録装置の組の個別の記述と及び前記記録装置の組に対する論理構造とを具備する前記メモリと；

前記メモリと及び命令のプログラムを実行する前記ネットワーク・インターフェースとに接続されたデータ・プロセッサであって；前記命令のプログラムは、

ネットワーク・インターフェースを通してドキュメントを具備するデータを受信するための論理と；

入力ドキュメントを識別するための仕様及び識別された入力ドキュメントを受信する一つ以上のトランザクションに従って、前記ドキュメントをパースするための論理と；及び

機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給するため

の論理とを含む前記データ・プロセッサとを具備することを特徴とする装置。

【請求項１４７】　論理構造のライブラリと、論理構造に対する模式図マップと及び参加者インターフェース記述を構築するために使用される論理構造を具備するドキュメントの定義を記録する前記データ・プロセッサによってアクセス可能なメモリに記録されたレポジトリを含むことを特徴とする請求項１４６に記載の装置。

【請求項１４８】　論理構造のライブラリと、論理構造に対する模式図マップと及び参加者インターフェース記述を構築するために使用される論理構造を具備するドキュメントの定義を記録するネットワーク・インターフェースを通してメモリに記録されるレポジトリへアクセスするための論理を含むことを特徴とする請求項１４６に記載の装置。

【請求項１４９】　前記機械可読仕様は、特定のトランザクションの識別子を記録するための論理構造と、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つとを含む参加者インターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項１４６に記載の装置。

【請求項１５０】　前記機械可読仕様は、参加者インターフェースの識別子を記録するための、及び前記参加者インターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項１４６に記載の装置。

【請求項１５１】　参加者インターフェース・ドキュメントの定義に従った前記ドキュメントは、特定のトランザクションの仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項１５０に記載の装置。

【請求項１５２】　前記記録装置はパースされたデータを具備することを特徴とする請求項１４６に記載の装置。

【請求項１５３】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタをコード化するキャラクタ・データと；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識別するマーク付けデータとを具備することを特徴とする請求項１５２に記載の装置。

【請求項１５４】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項１５３に記載の装置。

【請求項１５５】　前記仕様は、前記入力及び出力ドキュメントのうち少なくとも一つの論理構造によって識別される前記記録装置の組のうち少なくとも一つに対する前記翻訳情報を含み、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項１５４に記載の装置。

【請求項１５６】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項１５４に記載の装置。

【請求項１５７】　機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給するための論理は、処理アーキテクチャに従ったルーティング処理を含み、及び：

前記参加者インターフェースにおける前記入力及び出力ドキュメントの定義に応答して、記録装置の前記組に対応するデータ構造及び前記トランザクション処理の処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造とをコンパイルするためと、及び前記入力ドキュメントを前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令とをコンパイルするためのコンパイラを含むことを特徴とする請求項１４６に記載の装置。

【請求項１５８】　機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給するための論理は、処理アーキテクチャに従ったルーティング処理を含

み、及び：前記入力ドキュメントの少なくとも一部分を、前記処理アーキテクチャに従って読み取り可能である形式に翻訳するための論理を含むことを特徴とする請求項１４６に記載の装置。

【請求項１５９】　前記翻訳するための論理は、前記ルーティング処理の処理アーキテクチャに従った変数及び方法を含むプログラミング・オブジェクトを生成する段階を含むことを特徴とする請求項１５８に記載の装置。

【請求項１６０】　機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給するための論理は、前記入力ドキュメントの前記部分を前記識別されたトランザクションにルーティングするためのルータ（ｒｏｕｔｅｒ）を含むことを特徴とする請求項１４６に記載の装置。

【請求項１６１】　前記ルータは、ネットワーク・インターフェース上の前記入力ドキュメントを、前記識別されたトランザクションの一つを実行するノードに送信するための論理を含むことを特徴とする請求項１６０に記載の装置。

【請求項１６２】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項１４６に記載の装置。

【請求項１６３】　参加者インターフェースの仕様は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義に従ってドキュメントの定義を具備することを特徴とする請求項１６２に記載の装置。

【請求項１６４】　前記レポジトリは複数のトランザクションにおける使用のための標準化されたドキュメント・タイプを含み、及び前記入力及び出力ドキュメントの一つの前記定義は、レポジトリにおける標準化されたドキュメント・タイプへの参照を含むことを特徴とする請求項１４７に記載の装置。

【請求項１６５】　前記レポジトリは、ネットワークにおける参加者処理を識別するための標準化されたドキュメント・タイプを含むことを特徴とする請求項１４７に記載の装置。

【請求項１６６】　前記トランザクション処理は、複数の可変トランザクション処理アーキテクチャの一つを各々有し、及び前記入力ドキュメントの少なく

とも一部分を、前記個別のトランザクション処理の前記可変トランザクション処理アーキテクチャに従って読み取り可能な形式に翻訳するため、及び前記翻訳された部分を前記個別のトランザクション処理にルーティングするための論理を含むことを特徴とする請求項１４６に記載の装置。

【請求項１６７】　前記翻訳するための論理は、前記個別のトランザクション処理の前記可変トランザクション処理アーキテクチャに従って変数及び方法を含むプログラミング・オブジェクトを生成することを特徴とする請求項１６６に記載の装置。

【請求項１６８】　前記トランザクション処理の前記可変トランザクション処理アーキテクチャは、インターフェース記述言語に従った処理を具備することを特徴とする請求項１６６に記載の装置。

【発明の詳細な説明】

【０００１】

（発明の属する技術分野）

本発明は、ネットワークに接続された多様なクライアントの間でのトランザクション（取引）をサポートするシステム及びプロトコルに関し、詳細には、いろいろなアーキテクチャを有するプラットフォームの間での商取引をサポートするシステム及びプロトコルに関する。

【０００２】

（従来の技術）

インターネットやその他の通信ネットワークは、参加者が商品やサービスを売買する商取引を含む幅広く多様なトランザクションに使用されつつあるコンピュータプラットフォームと人との間の通信のための手段を提供している。インターネット上での商取引を促進させるために多くの努力がなされつつある。しかしながら、多くの競合する標準があるので、トランザクションを実行するためには、そのトランザクションへの関係者は、前もって、用いられるプロトコルに同意しなければならないし、しばしば、かかるトランザクションをサポートするためプラットフォームアーキテクチャのカスタムインテグレーション（専用の統合）を必要とする。同意した標準にコンパチブルでない特定のノード内への商業的プロセスは、他のノードとのインテグレーションのために相当の手直しを必要とするかも知れない。更に、一つの会社（Ｃｏｍｐａｎｙ）が１つの標準又は他の標準にコミットすると、その会社は標準化グループのトランザクション関係者に固定され、それ以外を排除する。

【０００３】

インターネット商取引の開発に遭遇した挑戦の概要が、「Ｅｃｏ Ｓｙｓｔｅｍ：An Ｉｎｔｅｒｎｅｔ Ｃｏｍｍｅｒｃｅ Ａｒｃｈｉｔｅｃｔｕｒｅ」Ｔｅｎｅｎｂａｕｍ 外著：Ｃｏｍｐｕｔｅｒ，Ｍａｙ １９９７；４８～５５ページに記載されている。

【０００４】

インターネット上での商取引を開放するため、アーキテクチャフレームワーク

の標準化が望ましい。かかる商業フレームワークをサポートするために開発されたプラットフォームは、ＩＢＭのＣｏｍｍｅｒｃｅ Ｐｏｉｎｔ；Ｍｉｃｒｏｓｏｆｔ Ｉｎｔｅｒｎｅｔ Ｃｏｍｍｅｒｃｅ Ｆｒａｍｅｗｏｒｋｓ；ＮｅｔｓｃａｐｅのＯＮＥ（Ｏｐｅｎ Ｎｅｔｗｏｒｋ Ｅｎｖｉｒｏｎｍｅｎｔ）；ＯｒａｃｌｅのＮＣＡ（Ｎｅｔｗｏｒｋ Ｃｏｍｐｕｔｉｎｇ Ａｒｃｈｉｔｅｃｔｕｒｅ）；及びＳｕｎ／ＪＡＶＡのソフトＪＥＣＦ（ＪＡＶＡ Ｅｌｅｃｔｒｏｎｉｃ Ｃｏｍｍｅｒｃｅ Ｆｒａｍｅｗｏｒｋ）を包含する。

【０００５】

これらの専有のフレームワークに加えて、ＣＯＲＢＡ ＩＩＯＰ（Ｃｏｍｍｏｎ Ｏｂｊｅｃｔ Ｒｅｑｕｅｓｔ Ｂｒｏｋｅｒ Ａｒｃｈｉｔｅｃｔｕｒｅ Ｉｎｔｅｒｎｅｔ ＯＲＢ Ｐｒｏｔｏｃｏｌ）に基づく、共通に配布されたオブジェクトモデルのようなプログラム技術が遂行されつつある。その共通配布のオブジェクトモデルの使用は、企業のシステムを、電子商取引のためのビジネスアプリケーションレベルで共同利用できるシステムにマイグレーション（交換）するのを簡単にしようとするためのものである。しかしながら、１つのフレームワークを用いる消費者又はビジネスは、異なるネットワーク上でトランザクションを実行することはできない。これが電子商取引システムの成長を制限している。

【０００６】

１つのフレームワークを実施する会社は、アプリケーションプログラミングインタフェース（ＡＰＩ）をもつであろうし、このＡＰＩは他のフレームワークをサポートするＡＰＩとは異なるものであろう。従って、会社にとって、共通のビジネスシステムインタフェースの採用を必要とせずに、相互にビジネスサービスにアクセスするのは大変に困難である。かかるＡＰＩレベルでのビジネスシステムインタフェースの開発は、関係者間でのかなりの協力が必要であるが、それはしばしば実際にはできない。

【０００７】

従って、通信ネットワーク内の多様なプラットフォーム間での対話（インタラクション）を促進するフレームワークを提供することが望ましい。かかるシステ

ムは、カスタムインテグレーションや産業上でのワイドな標準に事前同意を必要とせずに、取引のパートナー間での自発的な商売を促進するであろう。更に、かかるシステムは、ビジネスオートメーションへの前進を促進して、伝統的なシステムインテグレーションの時間やコストやリスクの多くを排除するであろう。

【０００８】

全般的に、専有の標準に基づいた、閉じた商業パートナーネットワークを、開放マーケットに置き換えられる電子商取引システムを提供するのが望ましい。

【０００９】

（発明の概要）

本発明は、ビジネスをカストマーや供給業者や商業パートナーにつなぐためのインフラストラクチャ（基盤＝インフラ）を提供する。本発明のインフラストラクチャの下では、複数の会社は相互に、自己定義のマシン読取り可能なドキュメント（例えば、パートナーの間で容易に理解できるＸＭＬ（拡張マークアップ言語）系のドキュメント）を用いて情報及びサービスを交換する。交換されるべきドキュメントを記述するドキュメント（本書ではビジネスインタフェース定義すなわちＢＩＤと呼ぶ）が、インターネット上で送られ、あるいは、ネットワークのメンバーに通信される。ビジネスインタフェース定義（ＢＴＤ）は、潜在的な商業パートナーに、会社が提供するサービスとそのサービスに通信する時に使用するためのドキュメントとを告げる。従って、代表的なビジネスインタフェース定義によって、カストマーは購買オーダ（その購買オーダを受取るパーティのＢＩＤで発行されたドキュメント定義に従う購買オーダ）を提出することによってオーダを行うことができる。供給業者は、ビジネスシステム管理在庫データのＢＩＤで発行されたドキュメント定義に従った在庫状況報告をダウンロードすることによってその有効性をチェックすることが可能になる。予め定義したマシン読取り可能なビジネスドキュメントの使用によって、エンタプライズ（企業）のアプリケーションにアクセスするためのより直観的な柔軟な方法が提供される。

【００１０】

商業ネットワーク内でのノードは、インタフェースのマシン読取り可能なスペシフィケーションから成る本発明によるトランザクションのためのインタフェー

スを、そのインタフェースのマシン読取り可能なスペシフィケーションのための解釈情報を含むマシン読取り可能なデータ構造とともに確立する。インタフェースのマシン読取り可能なスペシフィケーションは、入力ドキュメントの定義と出力ドキュメントの定義とを含んでおり、これらは、前記のノードがインタフェースとして動作するトランザクションプロセッサによって受入れられ、作られる。入力ドキュメントの定義及び出力ドキュメントの定義は、例えば、標準ＸＭＬ系ドキュメントに従った、記憶ユニットのセットの記述及び記憶ユニットのセットのためのロジック構造をそれぞれ包含する。本発明の種々の観点に従った解釈情報を含むマシン読取り可能なデータ構造は、入力及び出力ドキュメントの定義におけるロジック構造のためのデータタイプのスペシフィケーション（例えば、ストリングやアレイ等）、ロジック構造のためのコンテントモデル（例えば、可能な値のリスト）、及び／又は、オーダのリストの各エントリへの特定のロジック構造のための記憶ユニットの予め定義したセットをマップするデータ構造を含み、ロジック構造のセマンティック定義（例えば、ネームを生成するためのマッピングコード）を提供する。

【００１１】

本発明の他の観点によれば、インタフェースは、ネットワークの少なくとも１つのノードによってアクセスできるメモリ内のリポジトリ（これはロジック構造のライブラリ及びロジック構造の解釈情報を記憶する）を含む。このリポジトリは入力及び出力ドキュメントの定義のライブラリとインタフェースのスペシフィケーションのライブラリとネットワークの参加者のインタフェースノードのスペシフィケーションのライブラリを含むように拡張され得る。

【００１２】

従って、本発明のトランザクションフレームワークの参加者は、トランザクションに包含される複数のノード実行プロセスを含む、ネットワークの複数のノード間でトランザクションを実行する。本発明の方法は、トランザクションのためのインタフェースのマシン読取り可能なスペシフィケーションを記憶することを含み、スペシフィケーションは、入力ドキュメントの定義と出力ドキュメントの定義とを含む。入力及び出力ドキュメントの定義は、それぞれ、記憶ユニットの

セットの記述と記憶ユニットのセットロジック構造とを包含する。好ましいシステムでは、それらの定義は、標準のＸＭＬドキュメントタイプの定義、すなわちＤＴＤに従うやり方で、セマンティックマッピングとコンテントモデリングといくつかのエレメントデータタイピングとによって拡張されて、表現される。トランザクションの参加者は、通信ネットワークを通してドキュメントを含むデータを受取る。参加者は、トランザクションのために記憶されたスペシフィケーションに従ってそのドキュメントを解析して、そのトランザクションのための入力ドキュメントを識別する。ドキュメントの解析後、入力ドキュメントの少なくとも一部は、出力を提供するトランザクションプロセスへマシン読取り可能なフォーマットで提供される。出力ドキュメントは、記憶されたスペシフィケーションの出力ドキュメントの定義に基づいて、トランザクションプロセスの出力を構成するように形成される。出力ドキュメントは、通信ネットワーク上に送られて、代表的には、入力ドキュメントのソースに戻され、あるいは、トランザクションの特定のタイプのニーズに合わせた他のところに送られる。

【００１３】

したがって、ビジネスインターフェースの定義は、例えばＸＭＬによって特定されるドキュメント及び特別のノードにおけるトランザクション処理サービスのフロントエンド上で実行するプログラム間のギャップをブリッジする。これらのフロントエンドは、例えば、ＪＡＶＡ仮想マシンによって、或いはネットワークの両端のシステムの相互接続に対して与える他の共通のアーキテクチャによって実現される。ビジネスインターフェースの定義は、トランザクションプロトコルがビジネスインターフェースの定義書を用いてプログラムされる技術を提供する。このトランザクションのプロトコルに対するプログラムはドキュメント形式の詳細な公式仕様書によって与えられる。

【００１４】

トランザクションのインターフェースの機械読み取り仕様書はネットワークにおける他のプラットフォームにアクセス可能にされる。参加者のプラットフォームは、補完的なノードにおいて特定されるトランザクションインターフェースによる入力ドキュメント及び出力ドキュメントを設計する資源を含む。したがって

、参加者のノードは、補完的インターフェースのための入力ドキュメントの定義、及び補完的インターフェースのための出力ドキュメントの定義にアクセスする資源を含む。アクセスしている参加者ノードのためにストアされた仕様書は、その仕様書にストアされたインターフェースの入力ドキュメントの定義にある補完的なインターフェースの出力ドキュメントの定義の少なくとも一部を含むことによって確立される。

【００１５】

本発明の他の特徴によるインターフェースのストアされた仕様書を確立するプロセスは、リポジトリから機械読み取り可能な仕様書のエレメントにアクセスすることを含む。このリポジトリは論理構造、コンテントモデル、および論理構造のための概略マップのライブラリ、およびインターフェース記述書を構築するために用いられる論理構造を有するドキュメントの定義をストアする。ネットワークにおいてアクセス可能なリポジトリは、容易に共有されることができるインターフェースの記述の構築を容易にする。特別なプロセスに対して形成された入力ドキュメント及び補完的なプロセスによって、リターンとして期待される出力ドキュメント間のあらゆる相違は、ネットワーク上の通信及び特別な情報を表現する共通の論理構造に同意することによって容易に交渉される。

【００１６】

本発明の一つの特徴によるトランザクションのインターフェースの機械読み取り可能な仕様書は、ネットワークのメンバー間で共有されるインターフェースドキュメントの定義に従うドキュメントを含む。インターフェースドキュメントの定義は特別なトランザクションの識別子、及び定義の少なくとも一つ及び特別なトランザクション用の入出力ドキュメントの定義に対するレファレンスをストアするための論理構造を有する。すなわち、特別なサービスに対するインターフェース記述書は入出力ドキュメントの定義を実際に含む。代わりに、それはこのような定義のリポジトリにおける位置、またはネットワークのどこかへのポインタを含むことができる。

【００１７】

本発明の他の代替方法によると、機械読み取り可能な仕様書は、インターフェ

ースの識別子をストアするため、および仕様書の少なくとも１つとインターフェースによってサポートされる１以上のトランザクションの組の仕様書に対するレファレンスをストアするための論理構造を含むインタフェースドキュメントの定義に従うビジネスインターフェース定義ＢＩＤドキュメントを含む。各々のサポートされたトランザクションに対して、ドキュメントは定義の少なくとも１つ及び特別なトランザクションに対する入出力ドキュメントの定義に対するレファレンスを含む。

【００１８】

本発明の他の特徴によると、入出力ドキュメントの定義書に定義された記憶ユニットは文字データをエンコードするテキスト文字を含む分析されたデータ、および入出力ドキュメントの論理構造による記憶ユニットの組を識別するマークアップデータを有する。本発明の他の特徴によると、記憶ユニットの組の少なくとも１つは、自然言語を与える複数のテキスト文字をエンコードする。これは、これらのドキュメントの人間の読者や開発者によって入出力ドキュメントの定義の使用を容易にする。

【００１９】

本発明の他の特徴によると、入出力ドキュメントの仕様書は論理構造によって識別される記憶ユニットの組の少なくとも１つに対する翻訳情報を含む。１つの例における翻訳情報は、分析された文字の組に対する定義書をエンコードする。他の例では、翻訳情報はコンテントモデル仕様書のために提供する。例えば、カタログにおける記述を作るためにマップされたコードの多くのリストを載せるように特定の論理構造に要求する。あるシステムでは、ドキュメントの論理構造における記憶ユニットは特別な実効の必要性を満足するように分析されていないデータのセットを含む。

【００２０】

本発明の他の特徴によると、特別なプラットフォームにおけるトランザクションプロセスは、複数の変形トランザクション処理アーキテクチャの１つであるトランザクション処理アーキテクチャを有する。したがって、参加者ノードは、入力ドキョメントの少なくとも一部を、情報を利用するトランザクションの変形ト

ランザクション処理アーキテクチャに読取り可能なフォーマットに変換するための資源を含む。ドキュメント定義のエレメントは、トランザクションプロセスの変形トランザクション処理アーキテクチャによる変数及び方法を含む。トランザクションプロセスのフロントエンドとして働くＪＡＶＡ仮想マシンを有する参加者に対して、ドキュメントにおける特別なフィールドは、データを含むＪＡＶＡオブジェクトに翻訳されるばかりでなく、ＪＡＶＡオブジェクトに関連した機能を得て、設定する。本発明によってサポートすることができる他のトランザクション処理アーキテクチャは、Ｃｏｍｍｏｎ Ｏｂｊｅｃｔ Ｒｅｑｕｅｓｔ Ｂｒｏｋｅｒ Ａｒｃｈｉｔｅｃｔｕｒｅ CORBA, Ｃｏｍｐｏｎｅｎｔ Ｏｂｊｅｃｔ Ｍｏｄｅｌ COM, Ｏｎ－ｌｉｎｅ Ｔｒａｎｓａｃｔｉｏｎ Ｐｒｏｃｅｓｓｉｎｇ OLTR, およびＥｌｅｃｔｒｏｎｉｃ Ｄａｔａ Ｉｎｔｅｒｃｈａｎｇｅ ＥＤＩの意味におけるインターフェース記述言語に従う処理を含む。

【００２１】

本発明の他の特徴によると、複数のトランザクションにおいて使用されるドキュメント形式をストアするリポジトリが設けられる。特別なトランザクションのための機械読み取り可能な仕様書は、リポジトリにおけるドキュメント形式を参照することによって入出力ドキュメントの少なくとも１つを定義する。他の特徴によると、リポジトリに含まれるドキュメント形式は、ネットワークにおける参加者を識別するためのドキュメント形式を有する。このようなドキュメント形式は、参加者を識別し、その参加者によって支持されるサービスを特定し、且つこれらのサービスの各々に対する入出力ドキュメントを特定するため構造を提供する。

【００２２】

上述の方法に加えて、本発明は、ネットワークインターフェース、上述したように、トランザクションのためのインターフェースの機械読み取り可能な仕様書を含むデータと命令のプログラムをストアするためのメモリ、およびメモリとネットワークインターフェースに接続されたデータプロセッサを含むノード間のトランザクションを管理する装置を提供する。本装置にストアされた命令のプログ

ラムは、トランザクションにおける参加者のための、上述したプロセスを実行するロジックを含む。

【００２３】

本発明は、更に、トランザクション処理アーキテクチャによるトランザクション処理を実行するネットワークとデータ処理資源を含むシステム上で実行されるトランザクションのための参加者インタフェースを確立するための装置を提供する。本装置は、システムによって実効可能であり、特別のトランザクションにおける参加者のための参加者インターフェースの定義を構築するツールを提供するシステムによって、アクセス可能な媒体にストアされた命令のプログラムを含む。参加者インターフェースの定義は、入力ドキュメントの定義及び出力ドキュメント定義を含む。入出力ドキュメント定義は、記憶ユニットのセットのための論理構造における記憶ユニットのセットのそれぞれの機械読み取り可能な記述書を有し、そしてそれは、本発明の１つの特徴として、ＸＭＬドキュメント形式の定義に従うことができる。

【００２４】

また、本発明のこの形態による参加者インタフェースを設定する装置は、データ処理システムによってアクセス可能な媒体に記憶された命令のプログラムを含み、かつを入力されたドキュメントを対応するデータ構造に変換するためにシステムによって実行可能な命令をコンパイルすべく、かつトランザクション処理の出力を出力ドキュメントの記憶部及びロジック構造のセットに変換するためにシステムによって実行可能な命令をコンパイルすべくトランザクション処理アーキテクチャに順応する入力及び出力ドキュメントの論理構造及び記憶部のセットに対応しているデータ構造をコンパイルするために入力及び出力ドキュメントの定義に応答する。

【００２５】

好ましいシステムにおける参加者インタフェースの定義を構築するためのツールは、インタフェース記述を構築するために用いられる論理構造及び論理構造に対する解釈情報のライブラリを記憶するリポジトリからの及び／又は相補的ノードからの定義の構成要素をアクセスするためにシステムによって実行可能な命令

を含む。本発明の種々の形態によれば、リポジトリは、論理構造のライブラリだけでなく論理構造、及び参加者インタフェースを特定するためのフォーマットを備えているドキュメントの定義も含む。本発明のこの形態によれば、ツールは、参加者ノードの記述に関連して上述した技術によるビジネス・インタフェースの仕様を構築するために設けられる。本発明のこの形態によるインタフェースを構築するためのツール及びインタフェースをトランザクション処理アーキテクチャとの通信のために必要なリソースにコンパイルするためのツールは、１好ましいシステムにおける参加者ノードで実施され、かつ入力及び出力ドキュメントを定義するインタフェース記述に基づいてネットワークの使用が成長するときにインタフェース記述の開発及び最適化に利用される。

【００２６】

従って、本発明の別の形態は、参加者インタフェースの定義を構築するためのツール及びを上述した機能を実行するコンパイラを含む、メモリ及びにメモリに記憶された命令を実行するデータ・プロセッサを含む装置を提供する。

【００２７】

本発明の更に別の形態によれば、参加者インタフェース記述の使用は、マーケット・メーカ・ノードの動作を可能にする。そのようなノードにおいて、インタフェースによって支援されたトランザクション、及びそのようなトランザクションの各々の入力及び出力ドキュメントを識別する複数の参加者インタフェースの機械可読仕様を記憶することを具備する、トランザクションを管理する方法が提供される。上述したように、入力及び出力ドキュメントは、ＸＭＬ標準によるような、記憶部及び記憶部に対する論理構造のセットの各々の記述を備えている。更に、トランザクションの定義及び参加者インタフェースの定義は、全て、ＸＭＬ又は他の標準化ドキュメント表現言語に順応する技術により特定されたドキュメントを備えている。そのようなマーケット・メーカ・ノードでは、ドキュメントを備えているデータは、通信ネットワークで受信される。ドキュメントは、識別された入力ドキュメントを受入れる一つ以上のトランザクションにおいて入力ドキュメントを識別するために仕様書によりパーズされる。入力ドキュメントの少なくとも一部は、一つ以上の識別されたトランザクションに関連付けられるト

ランザクション処理に対して機械可読フォーマットで供給される。トランザクション処理に入力ドキュメントの少なくとも一部を供給する方法は、マーケット・メーカ・ノードにおける処理アーキテクチャによりルーティング処理を実行することを含む。本発明の一つの形態におけるルーティング処理は、特定の処理アーキテクチャにより実行され、かつ入力ドキュメントの少なくとも一部は、ルーティング処理の処理アーキテクチャのフォーマットに変換される。好ましい形態による変換は、ルーティング処理の処理アーキテクチャによる変数及び方法を含むプログラミング・オブジェクトを生成することを含む。

【００２８】

本発明の別の形態によれば、マーケット・メーカ・ノードは、また、リポジトリ構造を支持する。そこで、処理は、マーケット・メーカ・ノードに記憶されるリポジトリへのネットワークの参加者によるアクセスを許容するためにマーケット・メーカ・ノードに含まれる。上述したように、リポジトリは、トランザクション・インタフェース・ドキュメントを構築するために論理構造の定義、解釈情報、及び参加者ノード用ドキュメント定義を含みかつ参加者を識別するビジネス・インタフェース定義のインスタンス、及び各々の参加者によって実行されるトランザクションを含む。

【００２９】

ルーティング処理は、種々の代替により、ドキュメントが送られる処理の可変処理アーキテクチャへの入力ドキュメントの変換、遠隔処理ノードにネットワークでその最初のドキュメント・フォーマットで入力ドキュメントを送ること（ルーティング）、又はそのような処理の組合せを含む。代替において、ルーティング処理は、また、一つの入力ドキュメント定義により定義された入力ドキュメントを、入力ドキュメントを待機するために登録されている処理に対する異なるドキュメント仕様により定義された異なるドキュメントに変換するための技術を含む。

【００３０】

また、マーケット・メーカ・ノードは、ネットワーク・インタフェース、参加者インタフェースの仕様を含んでいる命令のプログラム及びデータを記憶してい

るメモリ、及びデータ・プロセッサを含む装置として本発明により提供される。ロジックは、上述したようなマーケット・メーカ処理を実行するために命令のプログラム等の形でデータ・プロセッサにより供給される。

【００３１】

従って、本発明は、入力及び出力ドキュメンテーションの仕様の共有化に基づぐ電子コマースに対する基礎を提供する。ドキュメントは、ＡＰＩｓをプログラム作成するよりも実施することがより簡単なビジネス・サービスをアクセスするための直感的かつ柔軟性のある方法を提供する。会社が既に大筋で同意しているならば、交換されるドキュメントによりそのような会社を相互接続することは、いつも異なるビジネス・システム・インタフェースよりも、更に容易である。加えて、そのようなドキュメントは、好ましい実施例ではヒューマン・リーダブル（人間が読取り可能な）・フォーマットで特定される。本発明によれば、ビジネス・インタフェースは、人間並びにコンピュータによって容易に解釈可能であるＸＭＬドキュメントのようなドキュメントで特定される。

【００３２】

本発明のドキュメント・ベース・トランザクション・アーキテクチャの利用は、ドキュメントのあらゆるソースと基本的には同じ方法で動作するパーズの使用を含み、ネットワークの各参加者から情報を抽出しかつ統合するためのカスタム・プログラムに対する多くの必要性を取り除く。そこで、会計、購入、製造、配送及び他の機能からの事業情報を統合することは、まず、各ソースを本発明によるアーキテクチャを有するドキュメントに変換し、次いでパーズされたデータ・ストリームを処理することによって達成することができる。マーケットに参加するシステムの各ノードは、その内部システムのフォーマット、並びにトランザクションにより交換のために特定されるドキュメントのフォーマットについて知ることが必要なだけである。そこで、電子コマース・ネットワークに参加することを希望する全ての他のノードのネイティブ（固有）・フォーマットを生成できることの必要性がない。

【００３３】

完全なビジネス統合のために、本発明は、ＸＭＬ構成要素、属性又はメタデー

タのような標準化論理構造のリポジトリを提供し、特定のコマース・コミュニティに対するセマンティック（意味）言語、異なるメタデータ記述間をマップするための手段、及びドキュメントを処理しかつ適切なアプリケーション及びサービスを呼出すためのサーバを設定する。本発明によるネットワークの基本的構築ブロックは、潜在的なトレーディング・ビジネス・パートナーにどのようなオンライン・サービスを会社が提供しかつサービスを呼出すためにどのドキュメントを用いるかを通告するインタフェース定義；及びトレーディング・コミュニティを生成するために内部及び外部ビジネス・サービスのセットを互いに結合するためにブリッジを供給するサーバを含む。サーバは、入力ドキュメントをパーズしかつ適切なサービスを呼出すために動作する。また、本発明によるサーバは、各々のホスト・システムのフォーマットへの及びそれらのフォーマットからの、受信しかつ送信するドキュメントのフォーマットからの変換タスクを取り扱う。そこで、トレーディング・パートナーは、交換したビジネス・ドキュメントの構造、内容及びシーケンシングだけに同意することが必要であり、アプリケーション・プログラマー・インタフェースの詳細に同意する必要はない。ドキュメントが処理される方法及びドキュメントの受信の結果としてもたらされるアクションは、まったくサービスを提供するビジネスによる。これは、システム・レベルからビジネス・レベルに統合を向上させる。それは、その内部技術実施、組織又は処理における変化にも係わらず、ビジネスにそのパートナーに対する明瞭かつ安定したインタフェースを提示させることができる。

【００３４】

ビジネス・インタフェース定義を構築しかつそのような記述によりサーバにコマースを管理させることができる全ての処理は、会社、サービス及び製品のようなビジネス記述プリミティブ、カタログ、購入注文、及びインボイスのようなビジネス・フォーム、並びに時間及び日付、場所及び商品の分類を含む、標準測定を含んでいる一般的なビジネス概念に対する情報モデルの共通のビジネス・ライブラリ、又はリポジトリによって容易に行われる。

【００３５】

本発明の他の形態は、以下に説明する図面、詳細な説明、及び特許請求の範囲

を考慮することにより理解することができるであろう。

【００３６】

（詳細な説明）

図面との関連で本発明について詳細に説明する。図１は、ビジネスインタフェース定義の使用に基づくマーケットメーカー及びマーケット参加者のネットワーク、該インタフェースの既述にしたがって特定された入力及び出力ドキュメントのやり取りのサポートを説明するものである。該ネットワークは、インターネット１９などの通信ネットワーク、あるいはその他のテレコミュニケーションまたはデータコミュニケーションネットワークを介して相互に接続されている、複数のノード１１～１８を含む。ノード１１～１９の各々は、ポータブルコンピュータ、デスクトップパーソナルコンピュータ、ワークステーション、システムのネットワーク、あるいは他のデータプロセシングリソースなどのコンピュータで構成されている。該ノードは、ビジネスインタフェース定義を記憶しておくためのメモリ、ネットワーク中の他のノードとの商取引トランザクションをサポートするトランザクションプロセスを実行するためのプロセッサ、及びそのようなサービスをサポートするために該プロセッサにより実行されるコンピュータプログラムを含んでいる。さらに、各ノードは、インターネット１９あるいは他のコミュニケーションネットワークを渡るコミュニケーションを提供するための、ネットワークインタフェースを含む。

【００３７】

図１の環境においては、ノード１１、１２、１３、１４、１６及び１８は、指定されたマーケット参加者である。マーケット参加者は、本発明にしたがって確立された商取引トランザクションにしたがって取引される商品またはサービスの需要者または供給者についてのリソースを含んでいる。

この例では、ノード１５及び１７は、マーケットメーカーノードである。マーケットメーカーノードは、ＢＩＤレジストリと呼ばれるビジネスインタフェース定義を登録するためのリソースを含んでいる。参加者は、ドキュメントをマーケットメーカーに送ることが可能であり、そこで該ドキュメントは識別され、そのようなドキュメントを入力として受け取るように登録されている適切な参加者に

発送される。マーケットメーカーはまた、ビジネスインタフェース定義の構築において使用するための共通ビジネスライブラリを構成する、標準フォームのリポジトリを維持することにより、商取引ネットワークを促進する。

【００３８】

この例では、マーケット参加者１８は、インターネット１９を介することなく、マーケットメーカー１７に直接接続されている。このマーケットメーカーへの直接接続は、インターネット１９などの公衆ネットワーク、及び、ローカルエリアネットワークのような私設接続あるいはノード１７と１８との間に示されているようなポイントートゥーポイント接続を実際の導入することにより、商取引トランザクションをサポートするネットワークの形態が極めて多様なものになり得る、ということを示すものである。実際のコミュニケーションネットワークはかなり多様であり、本発明による商取引トランザクションネットワークを確立するために使用するのに好適である。

【００３９】

図２は、本発明によるネットワークにおけるマーケット参加者のために確立されたビジネスインタフェース定義（ＢＩＤ）における、繰り込まれた構造の発見的なダイアグラムである。図２に示されているビジネスインタフェース定義は、ＸＭＬドキュメントタイプ定義ＤＴＤなどのドキュメント構造の形式定義にしたがって配列された記憶ユニット及び論理構造からなる、データ構造である。図２の構造は、パーティを識別するための第１の論理構造２００を含んでいる。論理構造２００と関連付けられているのは、名前を搬送するための繰り込まれた論理構造２０１、物理的アドレス２０２、ネットワークのアドレスあるいは位置２０３、及び一組のサービスのためのトランザクション２０４である。サービスセット中の各トランザクションについて、トランザクションＢＩＤ２０５、トランザクションＢＩＤ２０６、及びトランザクションＢＩＤ２０７を含む、インタフェース定義が提供されている。トランザクションＢＩＤ２０５などの各トランザクションＢＩＤの中には、名前２０８、サービスが実行されるネットワーク上の位置２０９、サービスにより実行される操作２１０、及び一組のタグにより示される入力ドキュメント２１１を含むための論理構造が提供されている。また、サー

ビスＢＩＤ２０５は、一組のタグにより示される出力ドキュメント２１２を含んでいる。一組の入力ドキュメント２１１は、入力ドキュメントビジネスインタフェース定義２１３、２１４、及び２１５を含む、サービスが応答するよう指示された、各入力ドキュメントについてのビジネスインタフェース定義を含んでいる。入力ドキュメントについてのビジネスインタフェース定義は、名前２１６、ドキュメントの記述を見出し得るネットワーク上の位置２１７、及びフィールドにより示された通りドキュメントにおいて搬送されるべきモジュール２１８を含んでいる。同様に、出力ドキュメントセット２１２は、出力ドキュメントＢＩＤ２１９、出力ドキュメントＢＩＤ２２０、及び出力ドキュメントＢＩＤ２２１を含む、出力ドキュメントについてのインタフェース定義を含んでいる。各出力ドキュメントＢＩＤについて、名前２２２、ネットワーク上または他の場所の位置２２３、及びドキュメントのモジュール２２４が特定されている。図２に示されている参加者についてのビジネスインタフェース定義は、各サービスの入力及び出力ドキュメントについて利用されるべき論理構造の実際の定義、あるいは、定義を見出し得る位置へのポインタまたは他のリファレンスを含んでいる。

【００４０】

好ましいシステムにおいては、図２のドキュメントは、ＸＭＬドキュメントタイプ定義ＤＴＤにおいて特定されるが、他のドキュメント定義アーキテクチャを使用することもでき、またドキュメントの依頼の翻訳に使用される論理構造についての翻訳情報を含んでいる。また、トランザクションＢＩＤ、入力ドキュメントＢＩＤ及び出力ドキュメントＢＩＤの各々は、ＸＭＬドキュメントタイプ定義にしたがって特定される。ＸＭＬタイプドキュメントは、マークアップデータ及びキャラクタデータを含む解析されたデータに基づくシステムの例である。マークアップデータは、ドキュメント内の論理構造を識別し、キャラクタデータの組は、論理構造の内容を識別する。さらに、解析されないデータは、様々な目的でドキュメント中を搬送され得る。例えば、ＷＷＷ．Ｗ３．ＯＲＧ／ＴＲ／１９９８／ＲＥＣ－ＸＭＬ－１９９８０２１０においてＷＣ３　ＸＭＬワーキンググループにより発行されたＥｘｔｅｎｓｉｂｌｅ　Ｍａｒｋ－ｕｐ　Ｌａｎｇｕａｇｅ　ＸＭＬ　１．０　ＲＥＣ－ＸＭＬ－１９９８０２１０の仕様書を参照のこと

【００４１】

かくして、例示したシステムにおいては、ネットワーク中の参加者ノードは、ビジネスシステム及びサービスを、ビジネスが受容し発生するＸＭＬコード化されたドキュメントと相互接続することにより、仮想企業を設立する。例えば、特定のサービスのビジネスインタフェース定義は、カタログ記入に対する要求のＢＩＤに合致するドキュメントを受け取った場合には、カタログ記入のＢＴＤに合致するドキュメントを返送することを確立する。また、購入注文のＢＩＤに合致するドキュメントを受け取り、それが受け取った端末にとって受け入れ可能なものである場合には、請求書のＢＩＤに合致するドキュメントを返送する。ネットワーク中のノードは、ネットワーク中の任意の所定のシステムの様々なトランザクションプロセシングアーキテクチャにしたがって確立されたローカルビジネスシステムにＸＭＬドキュメントが入る前に、ＸＭＬドキュメントを加工する。かくして、システムは、ＭＩＭＥーコード化されたＸＭＬドキュメントの組などの関連するドキュメントの組を解読し、これを解析して“マークアップメッセージ”のストリームを創作する。このメッセージは、例えば下記のようなイベントリスナーモデルを使用して、適切な用途及びサービスに発送される。

【００４２】

ビジネスサービス間で交換されるドキュメントは、より複雑なドキュメント定義が生成されうるビルディングブロックのリポジトリから生成されるＸＭＬ言語（共通のビジネス言語）を使用してエンコードされる。リポジトリは、会社、サービス、製品のようなビジネスディスクリプションプリミティブ、カタログ、購入注文、インボイスのようなビジネスフォーム、時間、日、位置のような標準の測定値、分類コード、および、ＸＭＬドキュメントの論理構造の解釈情報を与えるようなものを含む、ビジネスドメインに共通のファンクションおよび情報に焦点をあてた解釈情報のモジュールを記憶する。

【００４３】

ビジネスインターフェイス定義は、本発明に従うドキュメントを交換するインターフェイスを設計するために使用されるスキーマとしての役割を果たす更に高

レベルのドキュメントである。そうして、ビジネスインターフェイスは、ＸＭＬに従って記述されるドキュメントと、特定のノードにおいてトランザクション処理サービスのフロントエンドで実行されるプログラムとの間のギャップを埋める。このようなフロントエンドはＪＡＶＡ仮想マシーン、またはネットワーク中のシステムの内部接続を規定する他の共通アーキテクチャによって実施される。そうして、ビジネスインターフェイス定義は、ビジネスインターフェイス定義ドキュメントを使用してトランザクションプロトコルをプログラムするのに用いられる技術を提供する。トランザクションのプロトコルのためのプログラムは、ドキュメントタイプの詳細な公式の仕様によって規定されている。

【００４４】

ＸＭＬフォーマットに従うマーケット関係者のドキュメントに基づく一例のビジネスインターフェイス定義ＢＩＤは、以下に与えられる。マーケット関係者ＤＴＤは、連絡先および住所情報をサービスおよび財政情報のディスクリプションと関連付けて、マーケット関係者に関するビジネス情報を分類する。このビジネス情報は、名前、コード、住所、ビジネス組織を示すための専用分類法、ビジネスのタームへのポインタから成る。加えて、マーケット関係者ＤＴＤによって識別されるサービスは、その関係者が応答し、生成する入力及び出力ドキュメントを記載する。そうして、説明に役立つ注釈付きのＸＭＬで記述されるマーケット関係者ＤＴＤ、サービスＤＴＤ、トランザクションドキュメントＤＴＤのために一例の共通ビジネス言語を使用するスキーマを規定するドキュメントは、以下に続く。

【００４５】

Market Participant Sample

```
<!DOCTYPE SCHEMA SYSTEM "bidl.dtd">
<SCHEMA>
<H1>Market Participant Sample BID</H1>
<META
WHO.OWNS="Veo Systems"      WHO.COPYRIGHT="Veo Systems"
WHEN.COPYRIGHT="1998"       DESCRIPTION="Sample BID"
WHO.CREATED=""              WHEN.CREATED=""
WHAT.VERSION=""             WHO.MODIFIED=""
WHEN.MODIFIED=""            WHEN.EFFECTIVE=""
WHEN.EXPIRES=""             WHO.EFFECTIVE=""
WHO.EXPIRES="">
</META>

<PROLOG>
<XMLDECL STANDALONE="no"></XMLDECL>
<DOCTYPE NAME="market.participant">
<SYSTEM>markpart.dtd</SYSTEM></DOCTYPE>
</PROLOG>

<DTD NAME="markpart.dtd">
<H2>Market Participant</H2>
<H3>Market Participant</H3>
<ELEMENTTYPE NAME="market.participant">
<EXPLAIN><TITLE>A Market Participant</TITLE>
<SYNOPSIS>A business or person and its service interfaces.</SYNOPSIS>
<P>A market participant is a document definition that is created to describe a business and at
least one person with an email address, and it presents a set of pointers to service interfaces
located on the network. In this example, the pointers have been resolved and the complete BID
is presented here.</P></EXPLAIN>
<MODEL><CHOICE>
<ELEMENT NAME="business"></ELEMENT>
<ELEMENT NAME="person"></ELEMENT>
</CHOICE></MODEL></ELEMENTTYPE>
```

<H3>Party Prototype</H3>
<PROTOTYPE NAME="party">
<EXPLAIN><TITLE>The Party Prototype</TITLE></EXPLAIN>
<MODEL><SEQUENCE>
<ELEMENT NAME="party.name" OCCURS="+"></ELEMENT>
<ELEMENT NAME="address.set"></ELEMENT>
</SEQUENCE></MODEL>
</PROTOTYPE>

<H3>Party Types</H3>
<ELEMENTTYPE NAME="business">
<EXPLAIN><TITLE>A Business</TITLE>
<SYNOPSIS>A business (party) with a business number attribute.</SYNOPSIS>
<P>This element inherits the content model of the party prototype and adds a business number attribute, which serves as a key for database lookup. The business number may be used as a cross-reference to/from customer id, credit limits, contacts lists, etc.</P></EXPLAIN>
<EXTENDS HREF="party">
<ATTDEF NAME="business.number"><REQUIRED></REQUIRED></ATTDEF>
</EXTENDS>
</ELEMENTTYPE>

<H3>Person Name</H3>
<ELEMENTTYPE NAME="person">
<EXPLAIN><TITLE>A Person</TITLE></EXPLAIN>
<EXTENDS HREF="party">
<ATTDEF NAME="SSN"><IMPLIED></IMPLIED></ATTDEF>
</EXTENDS>
</ELEMENTTYPE>

<H3>Party Name</H3>
<ELEMENTTYPE NAME="party.name">
<EXPLAIN><TITLE>A Party's Name</TITLE>
<SYNOPSIS>A party's name in a string of character.</SYNOPSIS></EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>

```
<H3>Address Set</H3>
<ELEMENTTYPE NAME="address.set">
<MODEL><SEQUENCE>
<ELEMENT NAME="address.physical"></ELEMENT>
<ELEMENT NAME="telephone" OCCURS="*"></ELEMENT>
<ELEMENT NAME="fax" OCCURS="*"></ELEMENT>
<ELEMENT NAME="email" OCCURS="*"></ELEMENT>
<ELEMENT NAME="internet" OCCURS="*"></ELEMENT>
</SEQUENCE></MODEL>
</ELEMENTTYPE>

<H3>Physical Address</H3>
<ELEMENTTYPE NAME="address.physical">
<EXPLAIN><TITLE>Physical Address</TITLE>
<SYNOPSIS>The street address, city, state, and zip code.</SYNOPSIS></EXPLAIN>
<MODEL><SEQUENCE>
<ELEMENT NAME="street"></ELEMENT>
<ELEMENT NAME="city"></ELEMENT>
<ELEMENT NAME="state"></ELEMENT>
<ELEMENT NAME="postcode" OCCURS="?"></ELEMENT>
<ELEMENT NAME="country"></ELEMENT>
</SEQUENCE></MODEL>
</ELEMENTTYPE>

<H3>Street</H3>
<ELEMENTTYPE NAME="street">
<EXPLAIN><TITLE>Street Address</TITLE>
<SYNOPSIS>Street or postal address.</SYNOPSIS></EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>

<H3>City</H3>
<ELEMENTTYPE NAME="city">
<EXPLAIN><TITLE>City Name or Code</TITLE>
```

<P>The city name or code is a string that contains sufficient information to identify a city within a designated state.</P>
</EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>

<H3>State</H3>
<ELEMENTTYPE NAME="state">
<EXPLAIN><TITLE>State, Province or Prefecture Name or Code</TITLE>
<P>The state name or code contains sufficient information to identify a state within a designated country.</P></EXPLAIN>
<MODEL><STRING DATATYPE="COUNTRY.US.SUBENTITY"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Postal Code</H3>
<ELEMENTTYPE NAME="postcode">
<EXPLAIN><TITLE>Postal Code</TITLE>
<P>A postal code is an alphanumeric code, designated by an appropriate postal authority, that is used to identify a location or region within the jurisdiction of that postal authority. Postal authorities include designated national postal authorities.</P></EXPLAIN>
<MODEL><STRING DATATYPE="string"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Country</H3>
<ELEMENTTYPE NAME="country">
<EXPLAIN><TITLE>Country Code</TITLE>
<P>A country code is a two-letter code, designated by ISO, that is used to uniquely identify a country.</P></EXPLAIN>
<MODEL><STRING DATATYPE="country"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Network Addresses</H3>
<ELEMENTTYPE NAME="telephone">
<EXPLAIN><TITLE>Telephone Number</TITLE>

<P>A telephone number is a string of alphanumerics and punctuation that uniquely identifies a telephone service terminal, including extension number.</P></EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>

<H3>Fax</H3>
<ELEMENTTYPE NAME="fax">
<EXPLAIN><TITLE>Fax Number</TITLE>
<P>A fax number is a string of alphanumerics and punctuation that uniquely identifies a fax service terminal.</P>
</EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>

<H3>Email</H3>
<ELEMENTTYPE NAME="email">
<EXPLAIN><TITLE>Email Address</TITLE>
<P>An email address is a datatype-constrained string that uniquely identifies a mailbox on a server.</P></EXPLAIN>
<MODEL><STRING DATATYPE="email"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Internet Address</H3>
<ELEMENTTYPE NAME="internet">
<EXPLAIN><TITLE>Internet Address</TITLE>
<P>An Internet address is a datatype-constrained string that uniquely identifies a resource on the Internet by means of a URL.</P></EXPLAIN>
<MODEL><STRING DATATYPE="url"></STRING></MODEL>
</ELEMENTTYPE>

</DTD>
</SCHEMA>

Service Description Sample

```
<!DOCTYPE schema SYSTEM "bidl.dtd">
<SCHEMA>
<H1>Service Description Sample BID</H1>
<META
WHO.OWNS="Veo Systems"      WHO.COPYRIGHT="Veo Systems"
WHEN.COPYRIGHT="1998"       DESCRIPTION="Sample BID"
WHO.CREATED="*"             WHEN.CREATED="*"
WHAT.VERSION="*"            WHO.MODIFIED="*"
WHEN.MODIFIED="*"           WHEN.EFFECTIVE="*"
WHEN.EXPIRES="*"            WHO.EFFECTIVE="*"
WHO.EXPIRES="*">
</META>

<PROLOG>
<XMLDECL STANDALONE="no"></XMLDECL>
<DOCTYPE NAME="service">
<SYSTEM>service.dtd</SYSTEM></DOCTYPE>
</PROLOG>

<DTD NAME="service.dtd">
<H2>Services</H2>

<H3>Includes</H3>
<!-- INCLUDE><SYSTEM>comments.bim</SYSTEM></INCLUDE -->

<H3>Service Set</H3>
<ELEMENTTYPE NAME="service.set">
<EXPLAIN><TITLE>Service Set</TITLE>
<SYNOPSIS>A set of services.</SYNOPSIS></EXPLAIN>
<MODEL>
<ELEMENT NAME="service" OCCURS="+"></ELEMENT>
</MODEL></ELEMENTTYPE>

<H3>Services Prototype</H3>
<PROTOTYPE NAME="prototype.service">
```

<EXPLAIN><TITLE>Service</TITLE></EXPLAIN>
<MODEL><SEQUENCE>
<ELEMENT NAME="service.name"></ELEMENT>
<ELEMENT NAME="service.terms" OCCURS="+"></ELEMENT>
<ELEMENT NAME="service.location" OCCURS="+"></ELEMENT>
<ELEMENT NAME="service.operation" OCCURS="+"></ELEMENT>
</SEQUENCE></MODEL>
<!-- ATTGROUP><IMPLEMENTS
HREF="common.attrib"></IMPLEMENTS></ATTGROUP -->
</PROTOTYPE>

<H3>Service</H3>
<INTRO><P>A service is an addressable network resource that provides interfaces to specific operations by way of input and output documents.</P></INTRO>
<ELEMENTTYPE NAME="service">
<EXPLAIN><TITLE>Service</TITLE>
<P>A service is defined in terms of its name, the location(s) at which the service is available, and the operation(s) that the service performs.</P></EXPLAIN>
<MODEL><SEQUENCE>
<ELEMENT NAME="service.name"></ELEMENT>
<ELEMENT NAME="service.location"></ELEMENT>
<ELEMENT NAME="service.operation" OCCURS="+"></ELEMENT>
<ELEMENT NAME="service.terms"></ELEMENT>
</SEQUENCE></MODEL>
</ELEMENTTYPE>

<H3>Service Name</H3>
<ELEMENTTYPE NAME="service.name">
<EXPLAIN><TITLE>Service Name</TITLE>
<P>The service name is a human-readable string that ascribes a moniker for a service. It may be employed is user interfaces and documentation, or for other purposes.</P></EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>

<H3>Service Location</H3>

<ELEMENTTYPE NAME="service.location">
<EXPLAIN><TITLE>Service Location</TITLE>
<SYNOPSIS>A URI of a service.</SYNOPSIS>
<P>A service location is a datatype-constrained string that locates a service on the Internet by means of a URI.</P></EXPLAIN>
<MODEL><STRING DATATYPE="uri"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Service Operations</H3>
<INTRO><P>A service operation consists of a name, location and its interface, as identified by the type of input document that the service operation accepts and by the type of document that it will return as a result.</P></INTRO>
<ELEMENTTYPE NAME="service.operation">
<EXPLAIN><TITLE>Service Operations</TITLE>
<P>A service operation must have a name, a location, and at least one document type as an input, with one or more possible document types returned as a result of the operation.</P>
</EXPLAIN>
<MODEL><SEQUENCE>
<ELEMENT NAME="service.operation.name"></ELEMENT>
<ELEMENT NAME="service.operation.location"></ELEMENT>
<ELEMENT NAME="service.operation.input"></ELEMENT>
<ELEMENT NAME="service.operation.output"></ELEMENT>
</SEQUENCE></MODEL>
</ELEMENTTYPE>

<H3>Service Operation Name</H3>
<ELEMENTTYPE NAME="service.operation.name">
<EXPLAIN><TITLE>Service Operation Name</TITLE>
<P>The service operation name is a human-readable string that ascribes a moniker to a service operation. It may be employed in user interfaces and documentation, or for other purposes.</P></EXPLAIN>
<MODEL><STRING></STRING></MODEL>
</ELEMENTTYPE>

<H3>Service Operation Location</H3>

<INTRO><P>The service location is a network resource. That is to say, a URI.</P></INTRO>
<ELEMENTTYPE NAME="service.operation.location">
<EXPLAIN><TITLE>Service Operation Location</TITLE>
<SYNOPSIS>A URI of a service operation.</SYNOPSIS>
<P>A service operation location is a datatype-constrained string that locates a service operation on the Internet by means of a URL.</P></EXPLAIN>
<MODEL><STRING DATATYPE="url"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Service Operation Input Document</H3>
<INTRO><P>The input to a service operation is defined by its input document type. That is, the service operation is invoked when the service operation location receives an input document whose type corresponds to the document type specified by this element.</P>
<P>Rather than define the expected input and output document types in the market participant document, this example provides pointers to externally-defined BIDs. This allows reuse of the same BID as the input and/or output document type for multiple operations. In addition, it encourages parallel design and implementation.</P></INTRO>
<ELEMENTTYPE NAME="service.operation.input">
<EXPLAIN><TITLE>Service Operation Input</TITLE>
<SYNOPSIS>Identifies the type of the service operation input document.</SYNOPSIS>
<P>Service location input is a datatype-constrained string that identifies a BID on the Internet by means of a URI.</P>
</EXPLAIN>
<MODEL><STRING DATATYPE="url"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Service Operation Output Document Type</H3>
<INTRO><P>The output of a service operation is defined by its output document type(s). That is, the service operation is expected to emit a document whose type corresponds to the document type specified by this element.</P></INTRO>
<ELEMENTTYPE NAME="service.operation.output">
<EXPLAIN><TITLE>Service Operation Output</TITLE>
<SYNOPSIS>Identifies the type of the service operation output document.</SYNOPSIS>
<P>Service location output is a datatype-constrained string that identifies a BID on the Internet by means of a URI.</P>

```
</EXPLAIN>
<MODEL><STRING DATATYPE="url"></STRING></MODEL>
</ELEMENTTYPE>

<H3>Service Terms</H3>
<INTRO><P>This is a simple collection of string elements, describing the terms of an
agreement.</P></INTRO>
<ELEMENTTYPE NAME="service.terms">
<EXPLAIN><TITLE>Service Terms</TITLE>
<SYNOPSIS>Describes the terms of a given agreement.</SYNOPSIS>
</EXPLAIN>
<MODEL><STRING DATATYPE="string"></STRING></MODEL>
</ELEMENTTYPE>

</DTD>
</SCHEMA>
```

サービスＤＴＤスキーマは、共通ビジネス言語リポジトリのサービスタイプエレメントで以下のように拡張されうる。

```
<!ELEMENT service.type EMPTY>
<!ATTLIST service.type
        service.type.name (
        catalog.operator
        | commercial.directory.operator
        | eft.services.provider
        | escrower
        | fulfillment.service
        | insurer
        | manufacturer
        | market.operator
        | order.originator
        | ordering.service
        | personal.services.provider
```

```
| retailer
| retail.aggregator
| schema.resolution.service
| service.provider
| shipment.acceptor
| shipper
| van
| wholesale.aggregator
)#REQUIRED
%common.attrib;
>
```

上記のサービスタイプエレメントは、ビジネスインターフェイス定義によって保有される解釈情報、この例では、有効なサービスタイプのいずれか１つの識別を可能にするコンテントフォームを例示する。別の解釈情報は、例えばコンテントフォーム「ｕｒｌ」を含み、データタイプ「ｓｔｒｉｎｇ」で表されるエレメント＜Ｈ３＞Ｉｎｔｅｒｎｅｔ　Ａｄｄｒｅｓｓ＜／Ｈ３＞のような、データタイピングを含む。さらに別の解釈情報は、例えばファイル「ＣＯＵＮＴＲＹ．ＵＳ．ＳＵＢＥＮＴＩＴＹ」中の国に対するコードマッピングを含むエレメント＜Ｈ３＞Ｓｔａｔｅ＜／Ｈ３＞のような、リストのエレメントへのコードのマッピングを含む。

マーケット関係者ＤＴＤによって照会されるサービスディスクリプションは、サービスのコンペティションについてサービスが認めて生成するドキュメントを規定する。ベーシックサービスディスクリプションは、ＸＭＬドキュメントｔｒａｎｓａｃｔ．ｄｔｄとして以下に記載される。

ｔｒａｎｓａｃｔ．ｄｔｄは、インボイスのようなトランザクションディスクリプション、または値の交換のディスクリプションをモデル化する。このドキュメントタイプは、多くの用途を支援し、従って、トランザクションディスクリプションエレメントは、ユーザーがインボイス、業績、売りの申し出、取引価格の要望等を区別するのを可能にする属性を有する。交換は２人以上の関係者間で行われうるが、申し出人と相手の２人のみを登場させる。その各人は、上記に概説したマーケット関係者ＤＴＤに合致するドキュメントへのポインタによって表さ

れる。相手のポインタが、売りの申し出を受け入れるのは任意である。交換ディスクリプションは以下に挙げるｔｒａｎｐｒｉｍ．ｍｏｄに記述され、価格および小計を含む。交換ディスクリプションに続いて、全体としてトランザクションに適用される料金が備えられ、全体の料金が与えられなければならない。従って、この例のトランザクションディスクリプションスキーマドキュメントｔｒａｎｓａｃｔ．ｄｔｄは以下に表される。

```
<!-- transact.dtd Version: 1.0 -->
<!-- Copyright 1998 Veo Systems, Inc. -->

...

<!ELEMENT transaction.description (meta?, issuer.pointer,
        counterparty.pointer?, exhange.description+, general.charges?,
        net.total?)>
<!ATTLIST transaction.description
        transaction.type (invoice | pro.forma | offer.to.sell | order
                | request.for.quote | request.for.bid
                | request.for.proposal | response.to.request.for.quote
                | response.to.request.for.bid
                | response.to.request.for.proposal) "invoice"
        %common.attrib;
        %altrep.attrib;
        %nl.attrib;
>
```

代表マーケット関係者およびサービスＤＴＤは、上記の定義に従って生成され、以下の通りである。

マーケット関係者ＤＴＤ

```
<!ELEMENT business (party.name+, address.set) >
<!ATTLIST business business.number CDATA #REQUIRED
```

```
                >
    <!ELEMENT party.name (#PCDATA )>
    <!ELEMENT city (#PCDATA )>
    <!ELEMENT internet (#PCDATA )>
    <!ELEMENT country (#PCDATA )>
    <!ELEMENT state (#PCDATA )>
    <!ELEMENT email (#PCDATA )>
    <!ELEMENT address.physical  (street , city , state , postcode? , country) >
    <!ELEMENT telephone (#PCDATA )>
    <!ELEMENT person  (party.name+ , address.set) >
    <!ATTLIST person SSN  CDATA  #IMPLIED
                >
    <!ELEMENT fax (#PCDATA )>
    <!ELEMENT street (#PCDATA )>
    <!ELEMENT address.set  (address.physical , telephone* , fax* , email* , internet*) >
    <!ELEMENT postcode (#PCDATA )>
    <!ELEMENT market.participant  (business | person) >
```

サービスＤＴＤ

```
    <!ELEMENT service.location (#PCDATA )>
    <!ELEMENT service.terms (#PCDATA )>
    <!ELEMENT service.operation.name (#PCDATA )>
    <!ELEMENT service.operation  (service.operation.name , service.operation.location ,
    service.operation.input , service.operation.output) >
    <!ELEMENT service  (service.name , service.location , service.operation+ , service.terms) >
    <!ELEMENT service.operation.input (#PCDATA )>
    <!ELEMENT service.operation.location (#PCDATA )>
    <!ELEMENT service.name (#PCDATA )>
    <!ELEMENT service.set (service+ )>
    <!ELEMENT service.operation.output (#PCDATA )>
```

ｔｒａｎｓａｃｔ．ｄｔｄによって生成されるドキュメントの一例が続く。

```
<?xml version="1.0"?>
<!-- rorder.xml Version:  1.0 -->
<!-- Copyright 1998 Veo Systems, Inc. -->

<!DOCTYPE transaction.description SYSTEM "urn:x-
veosystems:dtd:cbl:transact:1.0:>
<transaction.description transaction.type="order">
<meta>
 <urn?urn:x-veosystems:doc:00023
 </urn>
        <thread.id party.assigned.by="reqorg">FRT876
        </thread.id>
</meta>
        <issuer.pointer>
                <xll.locator urllink="reqorg.xml">Customer
                Pointer
                </xll.locator>
        </issuer.pointer>
        <counterparty.pointer>
                <xll.locator urllink="compu.xml">Catalog entry owner
                pointer
                </xll.locator>
        </counterparty.pointer>
<exchange.description>
<line.item>
        <product.instance>
        <product.description.pointer>
                <xll.locator urllink="cthink.xml">Catalogue Entry    Pointer
                </xll.locator>
        </product.description.pointer>
        <product.specifics>
        <info.description.set>
<info.description>
 <xml.descriptor>
        <doctype>
```

```
        <dtd system.id="urn:x-veosystems:dtd:cbl:gprod:1.0"/>
        </doctype>
        <xml.descriptor.details>
        <xll.xptr.frag>DESCENDANT(ALL,os)STRING("Windows
95")
        </xll.xptr.frag>
        <xll.xptr.frag>DECENDANT(ALL.p.speed)STRING("200")
        </xll.xptr.frag>
        <xll.xptr.frag>DESCENDANT(ALL.hard.disk.capacity)
                STRING("4")
        </xll.xptr.frag>
        <xll.xptr.frag>DESCENDANT(ALL.d.size)STRING("14.1")
        </xll.xptr.frag>
        </xml.descriptor.details>
 </xml.descriptor>
</info.description>
        </info.description.set>
        </product.specifics>
<quantity>1
</quantity>
        </product.instance>
<shipment.coordinates.set>
<shipment.coordinates>
<shipment.destination>
                <address.set>
        <address.named>SW-1
        </address.named>
        <address.physical>
                <building.sublocation>208C</building.sublocation>
                <location.in.street>123
                </location.in.street>
                <street>Frontage Rd.
                </street>
                <city>Beltway
                </city>
```

```
                <country.subentity.us
                country.subentity.us.name="MD"/>
                <postcode>20000
                </postcode>
            </address.physical>
            <telephone>
                <telephone.number>617-666-2000
                </telephone.number>
                <telephone.extension>1201
                </telephone.extension>
            </telephone>
            </address.set>
        </shipment.destination>

        <shipment.special>No deliveries after 4 PM</shipment.special>
        </shipment.coordinates>
        </shipment.coordinates.set>
            <payment.set>
            <credit.card
            issuer.name="VISA"
            instrument.number="3787-812345-67893"
            expiry.date="12/97"
            currency.code="USD"/>
            <amount.group>
                <amount.monetary currency.code="USD">3975
                </amount.monetary>
            </amount.group>
            </payment.set>
        </line.item>
        </exhange.description>
        </transaction.description>
```

【００４６】

したがって、本発明は、マーケット参加者がそれ自身を識別し、入力ドキュメントのタイプおよび取引を行おうとしている出力ドキュメントのタイプを識別できるようにする技術を提供する。この種のドキュメントに担持された内容がこの取引での他のパーティーまたはローカルパーティーによって処理される特定の仕

方は、ビジネス関係の確立には関係しないし、取引を行うことにも関係しない。

【００４７】

図３は、本発明によるネットワークにおける参加者ノードの概略図である。図３に例示されたノードは、ポート３０１にて通信ネットワークに結合されるネットワークインターフェイス３００を含む。このネットワークインターフェイスは、ドキュメントパーサー３０１に結合される。パーサー３０１は、入力ドキュメントからのロジカルストラクチャーをトランスレータモジュール３０２へ供給し、トランスレータモジュール３０２は、入力ドキュメントをホスト取引システムによって使用できるようなフォームへと変換し、また、その逆に、ホストプロセッサの出力を送り先への送信のためにビジネスインターフェイス定義での出力ドキュメントフォームへと変換するためのものである。パーサー３０１およびトランスレータ３０２は、参加者モジュール３０３に記憶されたビジネスインターフェイス定義に応答する。

【００４８】

トランスレータ３０２からの出力データストラクチャーは、パーサー３０１によってシグナリングされるエベントと共に取引プロセスフロントエンド３０４へ供給される。１つの実施例におけるフロントエンド３０４は、ネットワークにおける種々なノードの間で通信するに適したＪＡＶＡバーチャルマシンまたはその他の類似のインターフェイスからなる。取引プロセスフロントエンド３０４は、パーサー３０１およびトランスレータ３０２によって指示されたエベントに応答して、入力データを、その参加者が結合されたエンタープライズシステムおよびネットワークにおける適当なファンクションへルーティングする。こうして、図３の例における取引プロセスフロントエンド３０４は、コマーシャルファンクション３０５、データベースファンクション３０６、アカウンティングおよびビリング３０７の如きその他のエンタープライズファンクションに結合され、また、パーサーによって指示されたエベントに応答するように設計された特定のエベントリスナーおよびプロセッサ３０８に結合される。

【００４９】

パーサー３０１は、ビジネスインターフェイス定義にしたがって特定される前

述の例におけるような購入注文またはその他のドキュメントを受け取り、ＪＡＶＡバーチャルマシンのためのＪＡＶＡエベントのセットの如きローカル取引処理アーキテクチャによって認識されるエベントのセットを生成する。

【００５０】

本発明のパーサーは、受信ドキュメントに基づくエベントをリスニングするプログラムから切り離される。受信ドキュメントにおける種々なマークアップピースまたは特定の仕様を満たす完全ドキュメントは、処理を開始させるリスニングファンクションのための命令として働く。こうして、リスニングプログラムは、ドキュメント情報に関連したビジネスロジックを実施する。例えば、アドレスエレメントに関連したプログラムは、データベースをチェックすることによってポスタルコードを確証するコードでありうる。これらのリスナーは、ドキュメントルーターで登録することによりエベントに加入する。ドキュメントルーターは、適切なエベントを、それらに興味のあるすべての加入者へと差し向けるものである。

【００５１】

例えば、前述したように特定された購入注文は、ドキュメントまたはその内容を注文エントリプログラムへ接続するであろうパーサーによって発生されるエベントをリスニングするプログラムによって監視されうる。その購入注文内のプロダクト記述の受領によりあるプログラムを使ってインベントリをチェックする。購入注文内のアドレス情報の受領により、あるプログラムを使ってデリバリのためのサービスの利用性をチェックする。ドキュメントにおけるバイヤー情報フィールドは、種々なプロセスを使って、クレジット価値について注文ヒストリをチェックし、または、プロモーションまたは消費者のアイデンティティを知ることに基づくような類似の処理をオファーすることができる。

【００５２】

コンプレックスリスナーは、プリミチブなリスナーのコンフィギュレーションとして生成されうる。例えば、ある購入注文リスナーは、前述したリストリスナーを含み、そのリストリスナーを使いうるし、また、リストメンバーは、それら自身に関して使われうる。リスナーがランするようなアプリケーションは、ネー

チブXMLプロセスまたはネーチブＪＡＶＡプロセスではないようであることに注意すべきである。これらの場合において、オブジェクトは、受信トランスアプリケーションによって必要とされるフォーマットへ変換される。そのアプリケーションが処理を完了するとき、その出力は、ネットワークにおける他のノードへの通信のためにXMLフォーマットへと変換し戻される。

【００５３】

前述したようなマーケット参加者ドキュメントタイプ記述および取引ドキュメントタイプ記述としては、ドキュメントにおけるロジックエレメントのためのスキマチックマッピングがあり、また、自然言語に基づくマークアップ言語がある。自然言語マークアップおよびXMLのその他の自然言語属性は、ビジネスインターフェイス定義、サービス記述および入力および出力ドキュメントの記述の仕様のためのXMLタイプマークアップ言語の使用を容易とする。

【００５４】

ビジネスインターフェイス定義を記憶することに加えて、参加者モジュール３０３は、入力ドキュメントにおけるロジカルストラクチャーに対応する取引プロセスフロントエンド３０４によって使用されるオブジェクトまたはその他のデータストラクチャーを編集したり、トランスレータ３０２を編集したりするのに使用されるコンパイラーを含む。こうして、ビジネスインターフェイス定義が参加者の関係する取引の変化につれて参加者によって変更されまたは更新されるとき、トランスレータ３０２およびパーサー３０１は、自動的にアップデートに保たれる。

【００５５】

好ましいシステムにおいては、ＪＡＶＡエベントのセットは、ＳＧＭＬのグローブモデル、主として、各エレメントについてプロパティセットによって拡張された標準のエレメントストラクチャーインフォーメーションセットに対応するコンパイラによって生成される（International Standard ISO／IEC 10179：1996（E），Information Technology――Processing Language――Document Style Semantic and Specificatio

ｎ Ｌａｎｇｕａｇｅ（ＤＳＳＳＬ））。ＸＭＬドキュメントをプロセスのためエベントのセットへと戻すのは、パーサー出力がインターナルデータストラクチャーとして維持されるような構文解析の通常モデルとは異なる。各ノードの取引処理フロントエンドによって使用するのに適したＪＡＶＡエベントまたはその他のプログラミングストラクチャーへＸＭＬドキュメントのエレメントを変換することにより、トレードされるドキュメントを利用するノードでのリッチファンクショナリティが可能とされる。

【００５６】

こうして、取引プロセスフロントエンド３０４は、システムにおける他のリスニングプログラムの知識なしにまたはインパクトなしに新しいリスナープログラムを加えることができるようにするパブリッシュおよびサブスクライブアーキテクチャーにて作動することができる。図３における各リスナー３０５、３０６、３０７、３０８は、フロントエンド３０４がエベントを指揮するキューを維持する。これにより、多数のリスナーがそれら自身のペースで併行してエベントを取り扱うことができるようになる。

【００５７】

さらにまた、本発明によれば、リスナーがランするようなアプリケーションは、入力ドキュメントのフォーマットに整合するネーチブＸＭＬファンクションまたはネーチブファンクションである必要はない。むしろ、これらのリスナーは、取引プロセスフロントエンド３０４がＪＡＶＡインターフェイスである場合には、ＪＡＶＡファンクションでありうし、または、独特な取引処理アーキテクチャーにしたがってランするファンクションでありうる。これらの場合において、オブジェクトは、受信アプリケーションによって必要とされるフォーマットへ変換されうる。リスナーのアプリケーションが終わるとき、その出力は、モジュール３０３におけるビジネスインターフェイス定義によって特定されるようなドキュメントのフォーマットへと変換し戻される。こうして、トランスレータ３０２は、組み合わせたドキュメントを出力として供給するため直接的にネットワークインターフェイス３００に結合される。

【００５８】

取引処理フロントエンドに結合されたリスナーは、入力ドキュメントのためのリスナー、入力ドキュメントの特定エレメントのためのリスナーおよび入力ドキュメントの特定のエレメントに記憶されている属性のためのリスナーを含みうる。これにより、入力ドキュメントをフィルタリングしそれに応答するための特定のノードでの取引プロセスの種々な融通性のある実施が可能となる。

【００５９】

図４は、図３のシステムのための入力ドキュメントを受信して処理するためのプロセスを例示している。こうして、プロセスは、ネットワークインターフェイスでドキュメントを受信することで始まる（ステップ４００）。パーサーは、ビジネスインターフェイス定義に応答してドキュメントタイプ（４０１）を識別する。ＸＭＬフォーマットにてドキュメントのためのＤＴＤを記憶しているビジネスインターフェイス定義を使用して、ドキュメントは構文解析される（ステップ４０２）。次に、ドキュメントのエレメントおよび属性は、ホストのフォーマットへと変換される（ステップ４０３）。この例では、ＸＭＬロジックストラクチャーは、ゲットおよびセットファンクションの如きデータに関連した方法とともにＸＭＬエレメントのデータを担持するＪＡＶＡオブジェクトへと変換される。次に、ホストオブジェクトは、ホスト取引処理フロントエンドへ転送される（ステップ４０４）。これらのオブジェクトは、パーサーおよびトランスレータによって指示されるエベントに応答してプロセスへルーティングされる。ドキュメントのエレメントを受け取るプロセスは、実行され、出力を発生する（ステップ４０５）。その出力は、ビジネスインターフェイス定義によって定められるような出力ドキュメントのフォーマットへと変換される（ステップ４０６）。この例では、トランスレーションは、ＪＡＶＡオブジェクトのフォームからＸＭＬドキュメントのフォームへと進む。最後に、出力ドキュメントは、ネットワークインターフェイスを通して送り先へと送信される（ステップ４０７）。

【００６０】

図５は、図３のシステムのためのイベント発生器／イベントリスナー機構の詳細な図である。一般に、図５に示されている手法はＪＡＶＡ　ＪＤＫ１．１イベントモデルの改良である。このモデルにおいて、３種類のオブジェクトが考慮さ

れる。第１種類のオブジェクトはイベントの発生に関する情報を含むイベントオブジェクトである。発生できるイベントの異なる全種類に対応して、イベントオブジェクトの種類の数が有る。第２種類のオブジェクトは、何かが起こった時に活動を監視しイベントオブジェクトを発生するイベント発生器である。第３は、イベント発生器により発生されたイベントオブジェクトを聞くイベントリスナーである。イベントリスナーは一般に、特定のウインドウ上のマウスクリックなどのような特定のイベント発生器を聞く。イベントリスナーはイベント発生器上に「イベントリスナーを加える」方法を呼ぶ。このモデルは図３の環境に適合させることができ、ＸＭＬドキュメントにより代表されるようなオブジェクトのグラフを歩きそして構文解析することに応答してオブジェクトが発生される。

【００６１】

図５に示されるシステムは一般的なＸＭＬパーサー５００を有する。このようなパーサーは標準の呼び戻しモデルを使用して実現できる。構文解析イベントが発生する時に、パーサーはアプリケーションオブジェクト内で特定の方法を呼出し、パラメータ内の適当な情報を渡す。従って、単一のアプリケーション５０１がパーサーと共に存在する。アプリケーションはパーサーが提供する情報をＸＭＬイベントオブジェクト内にパッケージし、そしてブロック５０２により示されるように、それを自身により識別された数だけのイベントリスナーに送る。イベント５０２の組は完全にパーサーから独立している。イベント５０２はどんな数のマシン上のどんな数のリスナーおよびどんな数のスレッドへにも供給されることができる。イベントは１つの代替形式においては、エレメント構造情報セット（ＥＳＩＳ）に基づいている。従って、これらはエレメントの開始および終了などのようなドキュメント構造または属性の認識のような重要な観点のリストからなる。ＸＭＬ（そしてＳＧＭＬ）パーサーは一般にＥＳＩＳ構造をパーサーがそのアプリケーションに戻るための情報のデフォルトセットとして使用する。

【００６２】

専門化されたＥＳＩＳリスナー５０３は、イベント５０２の組に結合される。このリスナー５０３はＥＳＩＳリスナーＡＰＩを実現し、そして１または複数からの発生器からの全てのＸＭＬイベントを聞く。１つのエレメントイベント発生

器５０４は専門化されたＥＳＩＳリスナーで、ＸＭＬイベント発生器でもある。そのリスナーは特定のタイプのエレメントに対するイベントにのみ興味があるオブジェクトである。例えば、ＨＴＭＬ環境においては、リスナーは指示されたリストのみ、すなわち、＜ＯＬ＞と＜／ＯＬ＞タグの間のドキュメントの部分のみに興味を有する。他の例では、リスナーは上の例のドキュメントから共通ビジネス言語に従って「ｐａｒｔｙ．ｎａｍｅ」エレメントまたは「ｓｅｒｖｉｃｅ．ｎａｍｅ」エレメントのみを聞き、エレメントのための図式的マッピングに一致するデータをエレメントが運ぶことを確実にするためにエレメントを処理し、そして受取ノードにおいて必要とされるプロセスに従って反応する。

【００６３】

これは、システムが、価格を合計するだけのようなドキュメントの特定部分のみを聞く小さいオブジェクトを有することを可能にする。リスナーが発生器からそれら自身を加えるまたは取除くことができるから、ＨＴＭＬドキュメントの例えば＜ＨＥＡＤ＞部分のみを聞くリスナーも存在できる。この理由およびＸＭＬドキュメントの高い反復性の理由により、高い目標的なコードを書くことおよび併発的なリスナーを書くことができる。例えば、＜ＯＬ＞リスナーは＜ＵＬ＞（指示されていないリスト）リスナーがその＜ＬＩ＞リスナーを設定する方法とは完全に分離して＜ＬＩ＞リスナーを設定することができる。代替方法として、図形ユーザインターフエイスを発生するリスナーおよび同じ入力を使用するデータベースを探索する別のリスナーを作成できる。従って、アプリケーションが一度に１つずつ検査する完成されたデータ構造とは反対に、ドキュメントはリスナーにより実行されるプログラムとして取扱われる。もし、アプリケーションがこの方法で書かれると、アプリケーションを実行するために全ドキュメントをメモリ内に持つ必要が無い。

【００６４】

イベント５０２の組に結合された次のリスナーは、属性フィルター５０５である。エレメントフィルタ５０４に似て属性フィルタ５０５はＥＳＩＳリスナーモデルに従い属性イベント発生器である。属性フィルタのためのリスナーはそれが興味を持つ属性を指定し、そして、指定された属性を持ついかなるエレメントの

ためのイベントを受取る。例えば、フォントマネージャは例えば＜Ｐ　ＦＯＮＴ＝“Ｔｉｍｅｓ　Ｒｏｍａｎ”／Ｐ＞のようなフォント属性を持つエレメントのためだけのイベントを受取るだろう。

【００６５】

エレメントイベント発生器５０４は、エレメントリスナー５０４Ａを専門化するためにそのようなエレメントオブジェクトを供給する。

【００６６】

属性イベント発生器５０５は属性イベントオブジェクトを属性リスナー５０５Ａへ供給する。同様に、属性オブジェクトは一のドキュメント形式から別へ属性を使用してＳＧＭＬ／ＸＭＬ変形する意味で「アーキテクチュア」に供給される。従って、５０５Ｂのアーキテクチュアは特定の名前を持った特定の属性を識別できるようにする。その定義された属性を持ったエレメントのみが出力ドキュメントの部分になり、そして出力ドキュメント内のエレメントの名前は、入力ドキュメント内の属性の値である。例えば、もしアーキテクチュア５０５ＢがＨＴＭＬであり、ストリングを、

＜ＰＵＲＣＨＡＳＥＳ　ＨＴＭＬ＝“ＯＬ”＞＜ＩＴＥＭ　ＨＴＭＬ＝“ＬＩ”＞＜ＮＡＭＥ　ＨＴＭＬ＝“Ｂ”＞ＳＴＵＦＦ＜／ＮＡＭＥ＞＜ＰＲＩＣＥ　ＨＴＭＬ＝“Ｂ”＞１２３＜／ＰＲＩＣＥ＞＜／ＩＴＥＭ＞＜／ＰＵＲＣＨＡＳＥ＞

翻訳すると、

＜ＯＬ＞＜ＬＩ＞ＳＴＵＦＦ＜／Ｂ＞＜Ｂ＞１２３＜／Ｂ＞＜／ＬＩ＞＜／ＯＬ＞

になり、正しいＨＴＭＬである。

【００６７】

イベント５０２の組に結合された次のモジュールはツリー（木）ビルダー５０６である。ツリービルダーはＸＭＬイベントのストリームを取り、下にあるドキュメントのツリー（木）表現を発生する。ツリービルダー５０６の一つの好ましい形式は、Ｗ３Ｃ（ｈｔｔｐ：／／ｗｗｗ．ｗ３．ｏｒｇ／ＴＲ／１９９８／ＷＤ－ＤＯＭ－１９９８０７２０／ｉｎｔｒｏｄｕｃｔｉｏｎ．ｈｔｍｌ）の仕様

に従ってドキュメントオブジェクトモデルＤＯＭオブジェクト５０７を発生する。イベントストリーム内のリスナーは大部分の要求を処理するのに使用できるが、ツリー形式はドキュメント回りの質問を支援するため、ノードを再順序付けるため、新ドキュメントの作成、例えばドキュメントを多数回構文解析するような、そこから同じイベントストリームを複数回発生できるメモリ内のデータ構造を支援するのに役立つ。専門化されたビルダー５０８は、特定の実施に適合するようなドキュメントの部分の特別なサブツリーを作るために、ツリービルダー５０６に結合できる。

【００６８】

入力するドキュメントに応答するのに加えて、ＸＭＬイベント５０２の他の源も供給されることができる。従って、イベントストリーム５１０はＤＯＭオブジェクトのツリー上を歩くことにより、そしてドキュメントが構文解析されいた時に作成された元のイベントストリームを再発生することにより発生される。これは、システムがドキュメントが数回構文解析されている外観を与えることを可能にする。

【００６９】

ツリーを歩きそしてイベントのストリームを発生するオブジェクトの観念は、ＤＯＭオブジェクトのツリーを越えて質問できるオブジェクトのいかなるツリーに一般化できる。従って、ＪＡＶＡウォーカー５１２は、ＪＡＶＡビーンコンポーネント５１３のツリーを歩くアプリケーションでもよい。ウォーカーは全ての公開されているフイールドおよび方法上を歩く。ウォーカーは終わりの無いサイクルに入り込まないことを確保するために既に訪問したオブジェクトの軌跡を記憶する。ＪＡＶＡイベント５１４はＪＡＶＡウォーカー５１２により発生されるイベントのタイプである。これは現在、オブジェクトから引き出すことのできる情報の種類の大部分を含む。これは、ＥＳＩＳのＪＡＶＡ等価物であり、そして、ＸＭＬに適用されたのと同じプログラム手法をＪＡＶＡオブジェクト一般に適用すること、特にＪＡＶＡビーンズにであるが、を可能にする。

【００７０】

ＪＡＶＡからＸＭＬイベント発生器５１５は、ＪＡＶＡリスナーとＪＡＶＡイ

ベント発生器から構成される。これは、ＪＡＶＡウォーカー５１２からイベントのストリーム５１４を受取り、そしてXMLドキュメントとしてＪＡＶＡオブジェクトを供給するために選択されたものを翻訳する。好ましい実施例において、イベント発生器５１５はＪＡＶＡビーンズＡＰＩを利用する。見られた各オブジェクトはエレメントとなり、エレメント名はクラス名とおなじである。そのエレメント内で、各埋め込まれた方法もまたエレメントとなり、その内容は方法を呼び起こすことにより戻された値である。もしそれがオブジェクトまたはオブジェクトの配列であると、これらは交互に歩かれる。

【００７１】

図６は、図５のフレームワーク上に構築された特定のアプリケーションの概略図である。このアプリケーションは、XMLドキュメント６００を取り込み、これをパーサ／発生器（構文解析ルーチン発生システム）６０１に適用する。ＥＳＩＳイベント６０２が発生され、且つ属性発生器６０３及びツリービルダー６０４に供給される。属性発生器は図５の発生器に対応する。この発生器５０５は、イベントを、例えば、XML入力からＨＴＭＬ出力に翻訳するための“アーキテクチャー”５０５Ｂに送る。これらのイベントは、ブロック６０５によって示される様に並列で処理され、且つリスナーによって処理される。リスナーの出力はドキュメントライター６０６に供給され、出力のためにXMLフォーマットに戻し翻訳される。従って、例えば、図６に示されるこのアプリケーションは、ＸＭＬドキュメントを取り入れ、或るフォームを有するＨＴＭＬドキュメントを出力する。このフォームは次にブラウザに送られ、その結果がXMLに戻し変換される。この運用のために、アーキテクチャコンセプトは、XMLからＨＴＭＬへのマッピングを提供する。図６に含まれる３つのアーキテクチャーは、テーブル及びリストの様なＨＴＭＬドキュメントの構造を与えるためのもの、ブラウザドキュメント上の入力フィールド用のラベルの様な表示されるべきドキュメントを特定する第２のもの、及び入力フィールド自体を記述する第３のものを含む。ＸＭＬドキュメント構造を維持するのに必要となるXMLドキュメントのエレメントは、ＨＴＭＬフォーム内で不可視のフィールドとなる。このことは、サーバに送り戻されるＨＴＴＰポストメッセージにクライアントが入れる情報からXMLド

キュメントを再構成する際に使用するのに有用である。これらのイベントを聴取するリスナーが、ＨＴＭＬドキュメント用のイベントを出力する。このドキュメントは、ドキュメントライターオブジェクトへ行く。ドキュメントライターオブジェクトはＸＭＬイベントを聴取し、それらをＸＭＬドキュメントに戻す。ドキュメントライターオブジェクトは、この例におけるアーキテクチャーを聴取している全てのエレメント発生器に対するリスナーである。

【００７２】

図５及び６に図示される処理モジュールの構成は、図３のシステム用のパーサ及びトランザクションプロセスフロントエンドの或る実施の形態を表している。理解できる様に、極めて柔軟性のあるインタフェースが与えられ、これによってダイバース（変更）トランザクションプロセスを、入力するＸＭＬドキュメント、又は他の構造化ドキュメントフォーマットに応答して、実行することができる。

【００７３】

図７は、ビジネスインタフェース定義ビルダーモジュール７００を含むことを除いて図３と類似のノードを図示している。従って、図７のシステムは、ネットワークインタフェース７０１、ドキュメントパーサ７０２、及びドキュメントトランスレータ７０３を含む。トランスレータ７０３は、その出力をトランザクション処理フロントエンド７０４へ出力する。このフロントエンドは、商業上の機能７０５、データベース７０６、企業機能７０７、及び他の一般リスナー及びプロセッサ７０８のようなリスニング機能に結合される。図７に図示される様に、ビジネスインタフェースビルダー７００は、ユーザインタェース、共通ビジネスライブラリＣＢＬリポジトリ、相補的ビジネスインタフェース定義をを読むためのプロセス、及びコンパイラを含む。ユーザインタフェースは、ビジネスインタフェース定義の構築における企てを、共通ビジネスライブラリリポジトリ及び相補的ビジネスインタフェースの定義を読み出す能力に依存して、補助するのに使用される。従って、相補的ビジネスインタフェースの定義の入力ドキュメントは、特定のトランザクションの出力ドキュメントとして特定することが出来、相補的ビジネスインタフェースの定義の出力ドキュメントがトランザクションプロセ

スの様な入力として特定することができる。同様な仕方で、トランザクションビジネスインタフェースの定義がＣＢＬレポジトリから選択された成分を使用して構成することがてきる。ＣＢＬリポジトリの使用が、上述の例示案（ｂｉｄ１）ドキュメントの様な標準化されたドキュメントフォーマット、及びネットワーク内の他の人間によって直ぐに採用することの出来るビジネスインタフェースの定義の構築に於ける論理構造及び解釈情報の使用を促進する。

【００７４】

ビジネスインタフェース定義ビルダー７００は、トランスレータ７０３、ホストトランザクション処理アーキテクチャに従ってトランスレータによって処理されるべきオブジェクトを発生し、パーシング機能７０２を管理するために使用されるコンパイラーも含む。

【００７５】

図８は、ビジネスインタフェース定義ビルダー７００内のリポジトリ内に記憶される論理構造を示す発見的図式である。従って、パーティビジネスインタフェース定義８００を表すリポジトリ記憶は、例えば、消費者ＢＩＤ８０１、カタログハウスＢＩＤ８０２、ウェアハウスＢＩＤ８０３、及びアクションハウスＢＩＤ８０４を含む。従って、オンラインマーケットにおける新たな参加者は、そのビジネスと最もよく適合する標準化されたＢＩＤの一つを、ベーシックインタフェース定義として、選択する。更に、リポジトリは一組のサービスビジネスインタフェース定義８０５を記憶する。例えば、注文エントリＢＴＤ８０６、注文トラッキングＢＩＤ８０７、注文履行ＢＩＤ８０８、及びカタログサービスＢＩＤ８０９を記憶することができる。マーケットにおける新たな参加者がビジネスインタフェースの定義を構築する際に、リポジトリ内に記憶される標準化されたサービスのビジネスインタフェース定義を選択することができる。

【００７６】

パーティ及びサービスＢＩＤに加えて、入力及び出力ドキュメントＢＩＤが、フィールド８１０によって指示されるリポジトリに記憶される。従って、購入注文ＢＩＤ８１１、請求書ＢＩＤ８１２、見積り要求ＢＩＤ８１３、製品入手可能リポートＢＩＤ８１４、及び注文状態ＢＩＤ８１５をリポジトリに記憶すること

が出来る。

【００７７】

リポジトリは、好適なシステムにおいては、ＸＭＬに従ったドキュメント形態の定義として特定されるビジネスインタフェースの定義に加えて、フィールド８１６によって指示される様なセマンティックマップの形態で解釈情報を記憶する。従って、重量８１７、通貨８１８、サイズ８１９、製品識別８２０、及び製品特徴８２１を特定するために使用されるセマンティックマップをリポジトリ内に記憶することができる。更に、解釈情報は、ドキュメントの論理構造内のデータ構造を分類分けすることに備える。

【００７８】

更に、ビジネスインタフェースの定義を構成する際に使用される論理構造は、ブロック８２２によって指摘されるリポジトリに記憶される。従って、アドレス情報８２３を与えるためのフォーム、価格情報８２４を与えるためのフォーム、契約上関係の期間８２５を与えるためのフォームを提供することができる。ネットワークが拡張される時、ＣＢＬリポジトリも同様に拡張し、標準化は、新たな参加者が加わること及びビジネスインタフェースの定義をより容易にする。

【００７９】

図９は、図７のシステムを使用するビジネスインタフェース定義を構築するプロセスを図示している。プロセスは、ＢＩＤビルダーグラフィカルインタフェースをユーザに表示することによって開始される（ステップ９００）。システムは、グラフィカルインタフェース（ステップ９０１）によって発生された参加者、サービス及びドキュメント情報を識別するユーザ入力を受け入れる。

【００８０】

次に、如何なる参照論理構造、解釈情報、ドキュメント定義、及び／又はサービス定義は、ユーザ入力に応答してグラフィカルユーザーインタフェースを介してリポジトリから検索される（ステップ９０２）。次のステップで、相補的ビジネスインタフェース定義又はビジネスインタフェースの定義の成分が、カスタム化された検索エンジン、ウェブブラウザ、又は他の方法によって、ユーザー入力を介して選択されたネットワーク内の他の参加者からアクセスされる（ステップ

９０３）。参加者に対するドキュメントの定義は、集められた情報を使用して発生される（ステップ９０４）。ドキュメントからホストへ及びホストからドキュメントへの各マッパーに対するトランスレータが、コンパイラによって発生される（ステップ９０５）。定義に対応するホストアーキテクチャデータ構造はコンパイラによって発生される（ステップ９０６）。そして発生されたビジネスインタフェースの定義は、ウェブサイト又は他の場所にポスト送付することにより、ネットワーク上にポスト送付され、ネットワーク内の他のノードにアクセス可能となるされる。（ステップ９０７）。

【００８１】

ビジネスインタフェースの定義は潜在的な商業パートナーに対して会社がオファーしたオンラインサービスを告げ、且つこれらのサービスを実施するのに使用するドキュメントを告げる。従って、サービスは、サービスが受け取り且つ発生するドキュメントによってビジネスインタフェースの定義で定められる。このことは、ＸＭＬサービスの定義の以下のフラグメントに例示されている。

【００８２】

```
<service>
<service.name>Order Service</service.name>
<service.location>ww.veosystems.com/order</service.location>
<service.op>
<service.op.name>Submit Order</service.op.name>
<service.op.inputdoc>www.commerce.net/po.dtd</service.op.inputdoc>
<service.op.outputdoc>
        www.veosystems.com/invoice.dtd</service.op.outputdoc>
</service.op>
<service.op>
<service.op.name>Track Order</service.op.name>
```

<service.op.inputdoc>www.commerce.net
/request.track.dtd<service.op.inputdoc>
<service.op.outputdoc>
www.veosystems.com/response.track.dtd<service.op.outputdoc>
</service.op>
</service>

【００８３】

このＸＭＬフラグメント（ｆｒａｇｍｅｎｔ）は、２つの処理からなるサービス、すなわち、１つは注文（ｏｒｄｅｒ）を取得する処理及びもう１つはそれに追従する処理を定義する。それぞれの定義は、もし正当な要求が特定のＷｅｂアドレスに提示されたなら、サービスを実行するための契約（ｃｏｎｔｒａｃｔ）又は約束（ｐｒｏｍｉｓｅ）を表現する。ここで注文サービスは、ローカルであるか、あるいはネットワーク上の産業全体のレジストリ中に記憶されたものでよいリポジトリ（ｒｅｐｏｓｉｔｏｒｙ）中に置かれている、標準の「ｐｏ．ｄｔｄ」ＤＴＤ（Ｄｏｃｕｍｅｎｔ　Ｔｙｐｅ　Ｄｅｆｉｎｉｔｉｏｎ）に適合する、入力されるドキュメント（ｄｏｃｕｍｅｎｔ）を必要とする。もし１つのノードがその注文を実行できるなら、そのノードは、定義がローカルである、カスタマイズされた「ｉｎｖｏｉｃｅ．ｄｔｄ」に適合するドキュメントを返すであろう。実際には、会社は、それが宣言するＸＭＬ仕様に適合する購入注文を提出できる誰とでも、ビジネスを行う約束をしている。事前の調整は不要である。

【００８４】

ＤＴＤは、所定の型のドキュメントのための公式の仕様すなわち文法であり、それは、要素、それらの属性、及びそれらが表示されなければならない状態を記述する。例えば、購入注文は、購入者及び販売者の名前及び住所、一組の製品説明、及び価格及び配達日のような関連する売買条件を典型的には含む。エレクトロニック・データ・インターチェンジ（Ｅｌｅｃｔｒｏｎｉｃ　Ｄａｔａ　Ｉｎｔｅｒｃｈａｎｇｅ，ＥＤＩ）では、例えばＸ１２　８５０仕様は、購入注文のために普通に使用されるモデルである。

【００８５】

リポジトリによって、多くのビジネス領域で共通の、再使用可能な意味の成分（ｒｅｕｓａｂｌｅ ｓｅｍａｎｔｉｃ ｃｏｍｐｏｎｅｎｔｓ）から、ＸＭＬドキュメントモデルを開発することが促進される。そのようなドキュメントは、たとえそれらの見かけが全く異なっていたとしても、それらの共通のメッセージ要素から理解することができる。これは、コモン・ビジネス・ライブラリ・リポジトリ（Ｃｏｍｍｏｎ Ｂｕｓｉｎｅｓｓ Ｌｉｂｒａｒｙ ｒｅｐｏｓｉｔｏｒｙ）の役割である。

【００８６】

コモン・ビジネス・ライブラリ・リポジトリは、以下を含む、一般的なビジネス概念のための情報モデルからなる。

・会社、サービス、及び製品のような、ビジネス記述プリミティブ（ｂｕｓｉｎｅｓｓ ｄｅｓｃｒｉｐｔｉｏｎ ｐｒｉｍｉｔｉｖｅ）、

・カタログ、購入注文、及びインボイスのようなビジネス形態、

・標準の測定値（ｓｔａｎｄａｒｄ ｍｅａｓｕｒｅｍｅｎｔｓ）、日時、位置、分類コード、である。

【００８７】

この情報は、会社がＸＭＬアプリケーションを迅速に開発するためにカスタマイズしかつ組み立てることができる、拡張可能で公開されたＸＭＬ構成ブロック（ＸＭＬ ｂｕｉｌｄｉｎｇ ｂｌｏｃｋ）として表現される。微少の（ａｔｏｍｉｃ）ＣＢＬ要素は、国、通貨、住所（ａｄｄｒｅｓｓ）、及び時間のための標準ＩＳＯコードのような、産業界のメッセージング規格及び慣習（ｉｎｄｕｓｔｒｙ ｍｅｓｓａｇｉｎｇ ｓｔａｎｄａｒｄｓ ａｎｄ ｃｏｎｖｅｎｔｉｏｎｓ）を実行する。低レベルのＣＢＬセマンティクス（ｓｅｍａｎｔｉｃｓ）は、ダブリン・コア（Ｄｕｂｌｉｎ Ｃｏｒｅ）のような、インターネット・リソースのための、提示されたメタデータ構成（ｍｅｔａｄａｔａ ｆｒａｍｅｗｏｒｋ）の分析からも得られる。

【００８８】

次のレベルの要素は、これらの構成ブロックを使用して、ＯＴＰ（Ｏｐｅｎ Ｔｒａｄｉｎｇ Ｐｒｏｔｏｃｏｌ）及びＯＢＩ（Ｏｐｅｎ Ｂｕｙｉｎｇ ｏ

ｎ　ｔｈｅ　Ｉｎｔｅｒｎｅｔ）のような、出現したインターネット規格で使用されるビジネス形態だけでなく、Ｘ１２　ＥＤＩ処理で使用されるもののような基本的なビジネス形態も実行する。

【００８９】

ＣＢＬの主眼は、全てのビジネス領域（会社、サービス、及び製品のような、ビジネス記述プリミティブ；カタログ、購入注文、及びインボイスのようなビジネス形態；標準の測定値、日時、位置、分類コード）に共通の機能及び情報にある。ＣＢＬは、可能な場合の意味論のための規格又は産業界の慣習に依存している（例えば、ヨーロッパでの「日／月／年」に対してアメリカでの「月／日／年」を指定する規則は、別個のＣＢＬモジュール中にコード化される）。

【００９０】

ＣＢＬは、アプリケーションを設計するために使用される言語である。それは、ＸＭＬの「ドキュメント・ワールド」と、ＪＡＶＡの「プログラミング・ワールド」又は他の処理プロセスアーキテクチャとの間のギャップを橋渡しするように設計されている。スキーマ（ｓｃｈｅｍａ）は、ドキュメントタイプの詳細な公式の仕様が種々の関連する形態がそこから生成されるようなマスター・ソース（ｍａｓｔｅｒ　ｓｏｕｒｃｅ）である、「ドキュメントによるプログラミング」の思想を具体化する。これらの形態は、ＣＢＬ、ＪＡＶＡオブジェクト、ＸＭＬのインスタンスと対応するＪＡＶＡオブジェクトとの間の変換プログラム、及びサポート・ドキュメントのためのＸＭＬ　ＤＴＤを含む。

【００９１】

ＣＢＬは、そこからシステムのほとんど全ての部分が自動的にコンパイラによって生成されるような単一のソースを作り出す。ＣＢＬは、それぞれの情報の要素及び属性と関連するセマンティクスの仕様を含むように、特定のドキュメントタイプの構造を公式に定義するために通常使用されるＳＧＭＬ／ＸＭＬを拡張することによって機能する。ＳＧＭＬ／ＸＭＬでの（ほとんどの）限られた組のキャラクタタイプを、任意の種類のデータタイプを宣言するために拡張することができる。

【００９２】

これは、「ｄａｔｅｔｉｍｅ」モジュールのためのＣＢＬ定義からのフラグメントである。

```
<!-- datetime.mod Version: 1.0-->
(!-- Copyright 1998 Veo Systems, Inc.--)
...
<! ELEMENT year (#PCDATA)>
<! ATTLIST year
    schema CDATA #FIXED "urn:x-veosystems:stds:iso:8601:3.8"
>

<! ELEMENT month (#PCDATA)>
<! ATTLIST month
    schema CDATA #FIXED "urn:x-veosystems:stds:iso:8601:3.12"
>

...
```

【００９３】

このフラグメントでは、ＥＬＥＭＥＮＴ”ｙｅａｒ”は、キャラクタデータ及び関連する”ｓｃｈｅｍａ”属性として定義され、キャラクタデータも、ＩＳＯ８６０１規格のセクション３．８になるように”ｙｅａｒ”のためのスキーマを定義する。

【００９４】

この”ｄａｔｅｔｉｍｅ”ＣＢＬモジュールは、実際は、スキーマＤＴＤのインスタンスとして定義される。最初に、モジュール名が定義される。次に、”ｄａｔｅｔｉｍｅ”要素の”ＹＥＡＲ”は、ＩＳＯ８６０１規格のセマンティクスに結合させられる。

【００９５】

```
<! DOCTYPE SCHEMA SYSTEM "schema.dtd">
<SCHEMA><H1>Date and Time Module</H1>

...
<ELEMNTTYPE NAME="year" DATATYPE="YEAR"><MODEL>
    <STRING
DATATYPE="YEAR"></STRING></MODEL>
<ATTDEF NAME=:schema:iso8601"DATATYPE="CDATA">
    <FIXED>3.8
Gregorian calendar</FIXED></ATTDEF></ELEMENTTYPE>

...
```

【００９６】

例のマーケットの参加者及び上記のサービスモジュールも、ＣＢＬリポジトリ中に記憶される。

【００９７】

図１０では、航空貨物受取証（Ａｉｒｂｉｌｌ）１０００が、一般的な購入注文ＤＴＤ１００１をカスタマイズし、出荷重量１００２についてのより具体的な情報を追加することによって定義されているところである。一般的な購入注文１００１は、住所、日時、通貨、及び販売者及び製品説明のためのＣＢＬモジュールから、広範囲にわたって、最初に組み立てられた。このように、ＣＢＬを使用することにより、ＸＭＬ商用アプリケーションのインプリメンテーションはかなり促進される。より重要なことに、ＣＢＬによって、商用アプリケーションを相互接続することが簡単になる。

【００９８】

ＣＢＬでは、ＸＭＬはスキーマにより拡張される。拡張によって、内容を簡単に確認できるように、ＸＭＬ要素に強調タイピング（ｓｔｒｏｎｇ－ｔｙｐｉｎｇ）が追加される。例えば、＜ＣＰＵ＿ｃｌｏｃｋ＿ｓｐｅｅｄ＞と呼ばれる要素は、一組の正当な値（ｖａｌｉｄ　ｖａｌｕｅ）：｛１００，１３３，１６６

，２００，２３３，２６６ Ｍｈｚ｝を有する整数として定義されることができる。スキーマは、情報をクラスの定義から簡単にインスタンス生成することができるように、クラスーサブクラスの階層も追加する。例えば、ラップトップを、ディスプレイのタイプ及びバッテリの寿命のような特徴のための追加のタグを有するコンピュータとして記述することができる。これら及び他の拡張により、ＸＭＬと伝統的なオブジェクト指向及び関係データモデルの間の自動翻訳だけでなく、データ入力も容易になる。

【００９９】

このようにして、完成されたＢＴＤは、参加者及び上記に概説したサービスの実際のインスタンス、ＤＴＤインスタンスの論理的構造に対応するＪＡＶＡビーンズ（ｂｅａｎｓ）、及びＸＭＬからＪＡＶＡに及びＪＡＶＡからＸＭＬに変換する変換コードのためのＤＴＤを作り出すコンパイラを通して実行される。他のシステムでは、オブジェクトを使用しやすくするために、ドキュメントも、ユーザインターフェース上のディスプレイのために、又はユーザによる印刷のために生成される。

【０１００】

例えば、市場参加者及び前述のサービスＤＴＤ、コンパイラによって生成されたＪＡＶＡビーンズは、（簡潔のためいくつかの修正を加えて）以下のように示される。

```
import com.veo.vsp.doclet.meta.Document;
public class AddressPhysical extends Document {
    public static final String DOC_TYPE = "address.physical";
     String mStreet;
     String mCity;
    public final static int AK = 0;
    public final static int AL = 1;
    public final static int AR = 2;
    public final static int AZ = 3;
    public final static int CA = 4;
    ...

    public final static int WI = 48;
    public final static int WV = 49;
    public final static int WY = 50;
     int mState;
     String mPostcode;
    public final static int AD = 51;
    public final static int AE = 52;
    public final static int AF = 53;
    public final static int AG = 54;
    public final static int AI = 55;
    public final static int AM = 56;
    ...

     int mCountry;
    public AddressPhysical(){
            super(DOC_TYPE);
    mStreet = new String();
    mCity = new String();
            this.mState = -1;
    mPostcode = null;
            this.mCountry = -1;
    }
    public AddressPhysical(String doc_type){
```

```
            super(doc_type);
        mStreet = new String();
        mCity = new String();
            this.mState = -1;
        mPostcode = null;
            this.mCountry = -1;
        }
        static public AddressPhysical initAddressPhysical(String iStreet,String iCity,int
iState,String iPostcode,int iCountry){
            AddressPhysical obj = new AddressPhysical();
            obj.initializeAll(iStreet, iCity, iState, iPostcode, iCountry);
            return obj;
        }

        public void initializeAll(String iStreet,String iCity,int iState,String iPostcode,int
iCountry){
            mStreet = iStreet;
            mCity = iCity;
            mState = iState;
            mPostcode = iPostcode;
            mCountry = iCountry;
        }
        public String getStreet(){
         return mStreet;
        }
        public String getStreetToXML(){
            if (getStreet() == null) return null;
            char [] c = getStreet().toCharArray();
            StringBuffer sb = new StringBuffer();
            for (int x = 0; x < c.length; x++){
             switch(c[x]){
             case '>':
              sb.append("&gt;");
              break;
             case '<':
```

```
                sb.append("&lt;");
                break;
              case '&':
                sb.append("&amp;");
                break;
              case '"':
                sb.append("&quot;");
                break;
              case '\':
                sb.append("&quot;");
                break;
              default:
                if (Character.isDefined(c[x]))
                  sb.append(c[x]);
              }
            }
            return sb.toString();
}
public void setStreet(String inp){
            this.mStreet = inp;
}
public void streetFromXML(String n){
            setStreet(n);
}
public String getCity(){
  return mCity;
}
public String getCityToXML(){
            if (getCity() == null) return null;
            char [] c = getCity().toCharArray();
            StringBuffer sb = new StringBuffer();
            for (int x = 0; x < c.length; x++){
              switch(c[x]){
              case '>':
                sb.append("&gt;");
```

```
                break;
              case '<':
                sb.append("&lt;");
                break;
              case '&':
                sb.append("&amp;");
                break;
              case '"':
                sb.append("&quot;");
                break;
              case '\'':
                sb.append("&quot;");
                break;
              default:
                if (Character.isDefined(c[x]))
                  sb.append(c[x]);
              }
            }
            return sb.toString();
}
public void setCity(String inp){
            this.mCity = inp;
}
public void cityFromXML(String n){
            setCity(n);
}
public int getState(){
 return mState;
}
public String getStateToXML(){
            switch (mState){
                        case AK: return "AK";
                        case AL: return "AL";
                        case AR: return "AR";
                        case AZ: return "AZ";
```

```
        }
  return null;
 }
 public void setState(int inp){
        this.mState = inp;
 }
 public void stateFromXML(String s){
        if (s.equals("AK")) mState = AK;
        else if (s.equals("AL"))mState = AL;
        else if (s.equals("AR"))mState = AR;
        else if (s.equals("AZ"))mState = AZ;
        ...

 }
 public String getPostcode(){
  return mPostcode;
 }
 public String getPostcodeToXML(){
        if (getPostcode() == null) return null;
        char [] c = getPostcode().toCharArray();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
         case '&':
          sb.append("&amp;");
          break;
         case '"':
```

```
                sb.append("&quot;");
                break;
              case '\':
                sb.append("&quot;");
                break;
              default:
                if (Character.isDefined(c[x]))
                  sb.append(c[x]);
            }
          }
          return sb.toString();
 }
 public void setPostcode(String inp){
          this.mPostcode = inp;
 }
 public void postcodeFromXML(String n){
          setPostcode(n);
 }
 public int getCountry(){
  return mCountry;
 }
 public String getCountryToXML(){
          switch (mCountry){
                  case AD: return "AD";
                  case AE: return "AE";
                  case AF: return "AF";
                  ...

          }
  return null;
 }
 public void setCountry(int inp){
          this.mCountry = inp;
 }
 public void countryFromXML(String s){
```

```
                if (s.equals("AD")) mCountry = AD;
                else if (s.equals("AE"))mCountry = AE;
                else if (s.equals("AF"))mCountry = AF;
                else if (s.equals("AG"))mCountry = AG;
                else if (s.equals("AI"))mCountry = AI;
                ..

        }
}

package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class AddressSet extends Document {
        public static final String DOC_TYPE = "address.set";
        AddressPhysical mAddressPhysical;
        String [] mTelephone;
        String [] mFax;
        String [] mEmail;
        String [] mInternet;
        public AddressSet(){
                super(DOC_TYPE);
                this.mAddressPhysical = new AddressPhysical();
        mTelephone = null;
        mFax = null;
        mEmail = null;
        mInternet = null;
        }
        public AddressSet(String doc_type){
                super(doc_type);
                this.mAddressPhysical = new AddressPhysical();
        mTelephone = null;
        mFax = null;
        mEmail = null;
        mInternet = null;
```

```
    }
    static public AddressSet initAddressSet(AddressPhysical iAddressPhysical,String []
iTelephone,String [] iFax,String [] iEmail,String [] iInternet){
        AddressSet obj = new AddressSet();
        obj.initializeAll(iAddressPhysical, iTelephone, iFax, iEmail, iInternet);
        return obj;
    }

    public void initializeAll(AddressPhysical iAddressPhysical,String [] iTelephone,String
[] iFax,String [] iEmail,String [] iInternet){
        mAddressPhysical = iAddressPhysical;
        mTelephone = iTelephone;
        mFax = iFax;
        mEmail = iEmail;
        mInternet = iInternet;
    }
    public AddressPhysical getAddressPhysical(){
     return mAddressPhysical;
    }
    public void setAddressPhysical(AddressPhysical inp){
        this.mAddressPhysical = inp;
    }
    public String [] getTelephone(){
     return mTelephone;
    }
    public String getTelephone(int index){
        if (this.mTelephone == null)
         return null;
        if (index >= this.mTelephone.length)
         return null;
        if (index < 0 && -index > this.mTelephone.length)
         return null;
        if (index >= 0) return this.mTelephone[index];
        return this.mTelephone[this.mTelephone.length + index];
    }
```

```
public String [] getTelephoneToXML(){
        String [] valArr = getTelephone();
        if (valArr == null) return null;
        String [] nvArr = new String[valArr.length];
        for (int z = 0; z < nvArr.length; z++){
        char [] c = valArr[z].toCharArray();
        StringBuffer st = new StringBuffer();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
         case '&':
          sb.append("&amp;");
          break;
         case '"':
          sb.append("&quot;");
          break;
         case '':
          sb.append("&quot;");
          break;
         default:
          if (Character.isDefined(c[x]))
           sb.append(c[x]);
         }
        }
        nvArr[z] = sb.toString();
        }
        return nvArr;
}
public void setTelephone(int index, String inp){
```

```
            if (this.mTelephone == null) {
                if (index < 0) {
                 this.mTelephone = new String[1];
                 this.mTelephone[0] = inp;
                } else {
                 this.mTelephone = new String[index + 1];
                 this.mTelephone[index] = inp;
                }
            } else if (index < 0) {
                        String [] newTelephone = new String[this.mTelephone.length + 1];
                        java.lang.System.arraycopy((Object)mTelephone, 0,
(Object)newTelephone, 0, this.mTelephone.length);
                        newTelephone[newTelephone.length - 1] = inp;
                        mTelephone = newTelephone;
                } else if (index >= this.mTelephone.length){
                        String [] newTelephone = new String[index + 1];
                        java.lang.System.arraycopy((Object)mTelephone, 0,
(Object)newTelephone, 0, this.mTelephone.length);
                        newTelephone[index] = inp;
                        mTelephone = newTelephone;
                } else {
                        this.mTelephone[index] = inp;
                }
        }
        public void setTelephone(String [] inp){
                this.mTelephone = inp;
        }
        public void telephoneFromXML(String n){
                setTelephone(-1, n);
        }
        public String [] getFax(){
         return mFax;
        }
        public String getFax(int index){
                if (this.mFax == null)
```

```
        return null;
      if (index >= this.mFax.length)
        return null;
      if (index < 0 && -index > this.mFax.length)
        return null;
      if (index >= 0) return this.mFax[index];
      return this.mFax[this.mFax.length + index];
}
public String [] getFaxToXML(){
      String [] valArr = getFax();
      if (valArr == null) return null;
      String [] nvArr = new String[valArr.length];
      for (int z = 0; z < nvArr.length; z++){
      char [] c = valArr[z].toCharArray();
      StringBuffer st = new StringBuffer();
      StringBuffer sb = new StringBuffer();
      for (int x = 0; x < c.length; x++){
       switch(c[x]){
       case '>':
        sb.append("&gt;");
        break;
       case '<':
        sb.append("&lt;");
        break;
       case '&':
        sb.append("&amp;");
        break;
       case '"':
        sb.append("&quot;");
        break;
       case '':
        sb.append("&quot;");
        break;
       default:
        if (Character.isDefined(c[x]))
```

```
                sb.append(c[x]);
                }
                }
                nvArr[z] = sb.toString();
                }
                return nvArr;
        }
        public void setFax(int index, String inp){
            if (this.mFax == null) {
                if (index < 0) {
                 this.mFax = new String[1];
                 this.mFax[0] = inp;
                } else {
                 this.mFax = new String[index + 1];
                 this.mFax[index] = inp;
                }
            } else if (index < 0) {
                        String [] newFax = new String[this.mFax.length + 1];
                        java.lang.System.arraycopy((Object)mFax, 0, (Object)newFax, 0,
this.mFax.length);
                        newFax[newFax.length - 1] = inp;
                        mFax = newFax;
                } else if (index >= this.mFax.length){
                        String [] newFax = new String[index + 1];
                        java.lang.System.arraycopy((Object)mFax, 0, (Object)newFax, 0,
this.mFax.length);
                        newFax[index] = inp;
                        mFax = newFax;
                } else {
                        this.mFax[index] = inp;
                }
        }
        public void setFax(String [] inp){
                this.mFax = inp;
        }
```

```
public void faxFromXML(String n){
        setFax(-1, n);
}
public String [] getEmail(){
 return mEmail;
}
public String getEmail(int index){
        if (this.mEmail == null)
         return null;
        if (index >= this.mEmail.length)
         return null;
        if (index < 0 && -index > this.mEmail.length)
         return null;
        if (index >= 0) return this.mEmail[index];
        return this.mEmail[this.mEmail.length + index];
}
public String [] getEmailToXML(){
        String [] valArr = getEmail();
        if (valArr == null) return null;
        String [] nvArr = new String[valArr.length];
        for (int z = 0; z < nvArr.length; z++){
        char [] c = valArr[z].toCharArray();
        StringBuffer st = new StringBuffer();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
         case '&':
          sb.append("&amp;");
          break;
```

```
                    case '"':
                     sb.append("&quot;");
                     break;
                    case '':
                     sb.append("&quot;");
                     break;
                    default:
                     if (Character.isDefined(c[x]))
                      sb.append(c[x]);
                    }
                   }
                   nvArr[z] = sb.toString();
                   }
                   return nvArr;
                }
                public void setEmail(int index, String inp){
                    if (this.mEmail == null) {
                        if (index < 0) {
                         this.mEmail = new String[1];
                         this.mEmail[0] = inp;
                        } else {
                         this.mEmail = new String[index + 1];
                         this.mEmail[index] = inp;
                        }
                    } else if (index < 0) {
                                String [] newEmail = new String[this.mEmail.length + 1];
                                java.lang.System.arraycopy((Object)mEmail, 0, (Object)newEmail,
0, this.mEmail.length);
                                newEmail[newEmail.length - 1] = inp;
                                mEmail = newEmail;
                           } else if (index >= this.mEmail.length){
                                String [] newEmail = new String[index + 1];
                                java.lang.System.arraycopy((Object)mEmail, 0, (Object)newEmail,
0, this.mEmail.length);
                                newEmail[index] = inp;
```

```
                mEmail = newEmail;
            } else {
                this.mEmail[index] = inp;
            }
        }
        public void setEmail(String [] inp){
            this.mEmail = inp;
        }
        public void emailFromXML(String n){
            setEmail(-1, n);
        }
        public String [] getInternet(){
         return mInternet;
        }
        public String getInternet(int index){
            if (this.mInternet == null)
             return null;
            if (index >= this.mInternet.length)
              return null;
            if (index < 0 && -index > this.mInternet.length)
              return null;
            if (index >= 0) return this.mInternet[index];
            return this.mInternet[this.mInternet.length + index];
        }
        public String [] getInternetToXML(){
            String [] valArr = getInternet();
            if (valArr == null) return null;
            String [] nvArr = new String[valArr.length];
            for (int z = 0; z < nvArr.length; z++){
            char [] c = valArr[z].toCharArray();
            StringBuffer st = new StringBuffer();
            StringBuffer sb = new StringBuffer();
            for (int x = 0; x < c.length; x++){
             switch(c[x]){
             case '>':
```

```
                sb.append("&gt;");
                break;
              case '<':
                sb.append("&lt;");
                break;
              case '&':
                sb.append("&amp;");
                break;
              case '"':
                sb.append("&quot;");
                break;
              case '"':
                sb.append("&quot;");
                break;
              default:
                if (Character.isDefined(c[x]))
                  sb.append(c[x]);
              }
            }
            nvArr[z] = sb.toString();
            }
            return nvArr;
}
public void setInternet(int index, String inp){
    if (this.mInternet == null) {
        if (index < 0) {
            this.mInternet = new String[1];
            this.mInternet[0] = inp;
        } else {
            this.mInternet = new String[index + 1];
            this.mInternet[index] = inp;
        }
    } else if (index < 0) {
                String [] newInternet = new String[this.mInternet.length + 1];
```

```
                        java.lang.System.arraycopy((Object)mInternet, 0,
(Object)newInternet, 0, this.mInternet.length);
                        newInternet[newInternet.length - 1] = inp;
                        mInternet = newInternet;
                } else if (index >= this.mInternet.length){
                        String [] newInternet = new String[index + 1];
                        java.lang.System.arraycopy((Object)mInternet, 0,
(Object)newInternet, 0, this.mInternet.length);
                        newInternet[index] = inp;
                        mInternet = newInternet;
                } else {
                        this.mInternet[index] = inp;
                }
        }
        public void setInternet(String [] inp){
                this.mInternet = inp;
        }
        public void internetFromXML(String n){
                setInternet(-1, n);
        }
}

package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class Business extends Party {
        public static final String DOC_TYPE = "business";
         String aBusinessNumber;
        public Business(){
                super(DOC_TYPE);
        aBusinessNumber = new String();
        }
        public Business(String doc_type){
                super(doc_type);
       aBusinessNumber = new String();
```

```
    }
    static public Business initBusiness(String iBusinessNumber,String [] 
iPartyName,AddressSet iAddressSet){
        Business obj = new Business();
        obj.initializeAll(iBusinessNumber, iPartyName, iAddressSet);
        return obj;
    }

    public void initializeAll(String iBusinessNumber,String [] iPartyName,AddressSet 
iAddressSet){
        aBusinessNumber = iBusinessNumber;
        super.initializeAll(iPartyName, iAddressSet);
    }
    public String getBusinessNumber(){
     return aBusinessNumber;
    }
    public String getBusinessNumberToXML(){
        if (getBusinessNumber() == null) return null;
        char [] c = getBusinessNumber().toCharArray();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
         case '&':
          sb.append("&amp;");
          break;
         case '"':
          sb.append("&quot;");
          break;
         case '\':
```

```
                    sb.append("&quot;");
                    break;
                  default:
                   if (Character.isDefined(c[x]))
                     sb.append(c[x]);
                  }
                 }
                 return sb.toString();
        }
        public void setBusinessNumber(String inp){
                this.aBusinessNumber = inp;
        }
        public void businessNumberFromXML(String n){
                setBusinessNumber(n);
        }
}


import com.veo.vsp.doclet.meta.Document;
public class Party extends Document {
        public static final String DOC_TYPE = "party";
        String [] mPartyName;
        AddressSet mAddressSet;
        public Party(){
                super(DOC_TYPE);
        mPartyName = new String[0];
                this.mAddressSet = new AddressSet();
        }
        public Party(String doc_type){
                super(doc_type);
        mPartyName = new String[0];
                this.mAddressSet = new AddressSet();
        }
        static public Party initParty(String [] iPartyName,AddressSet iAddressSet){
                Party obj = new Party();
```

```
        obj.initializeAll(iPartyName, iAddressSet);
        return obj;
    }

    public void initializeAll(String [] iPartyName,AddressSet iAddressSet){
        mPartyName = iPartyName;
        mAddressSet = iAddressSet;
    }
    public String [] getPartyName(){
     return mPartyName;
    }
    public String getPartyName(int index){
        if (this.mPartyName == null)
         return null;
        if (index >= this.mPartyName.length)
          return null;
        if (index < 0 && -index > this.mPartyName.length)
          return null;
        if (index >= 0) return this.mPartyName[index];
        return this.mPartyName[this.mPartyName.length + index];
    }
    public String [] getPartyNameToXML(){
        String [] valArr = getPartyName();
        if (valArr == null) return null;
        String [] nvArr = new String[valArr.length];
        for (int z = 0; z < nvArr.length; z++){
        char [] c = valArr[z].toCharArray();
        StringBuffer st = new StringBuffer();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
```

```
                    sb.append("&lt;");
                    break;
                  case '&':
                    sb.append("&amp;");
                    break;
                  case '"':
                    sb.append("&quot;");
                    break;
                  case '':
                    sb.append("&quot;");
                    break;
                  default:
                    if (Character.isDefined(c[x]))
                      sb.append(c[x]);
                  }
                 }
                 nvArr[z] = sb.toString();
                 }
                 return nvArr;
        }
        public void setPartyName(int index, String inp){
           if (this.mPartyName == null) {
                 if (index < 0) {
                  this.mPartyName = new String[1];
                  this.mPartyName[0] = inp;
                 } else {
                  this.mPartyName = new String[index + 1];
                  this.mPartyName[index] = inp;
                 }
           } else if (index < 0) {
                        String [] newPartyName = new String[this.mPartyName.length + 1];
                        java.lang.System.arraycopy((Object)mPartyName, 0,
(Object)newPartyName, 0, this.mPartyName.length);
                        newPartyName[newPartyName.length - 1] = inp;
                        mPartyName = newPartyName;
```

```
                } else if (index >= this.mPartyName.length){
                        String [] newPartyName = new String[index + 1];
                        java.lang.System.arraycopy((Object)mPartyName, 0,
(Object)newPartyName, 0, this.mPartyName.length);
                        newPartyName[index] = inp;
                        mPartyName = newPartyName;
                } else {
                        this.mPartyName[index] = inp;
                }
        }
        public void setPartyName(String [] inp){
                this.mPartyName = inp;
        }
        public void partyNameFromXML(String n){
                setPartyName(-1, n);
        }
        public AddressSet getAddressSet(){
         return mAddressSet;
        }
        public void setAddressSet(AddressSet inp){
                this.mAddressSet = inp;
        }
}

package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class Person extends Party {
        public static final String DOC_TYPE = "person";
         String aSSN;
        public Person(){
                super(DOC_TYPE);
        aSSN = null;
        }
        public Person(String doc_type){
```

```
        super(doc_type);
    aSSN = null;
    }
    static public Person initPerson(String iSSN,String [] iPartyName,AddressSet
iAddressSet){
        Person obj = new Person();
        obj.initializeAll(iSSN, iPartyName, iAddressSet);
        return obj;
    }

    public void initializeAll(String iSSN,String [] iPartyName,AddressSet iAddressSet){
        aSSN = iSSN;
        super.initializeAll(iPartyName, iAddressSet);
    }
    public String getSSN(){
     return aSSN;
    }
    public String getSSNToXML(){
        if (getSSN() == null) return null;
        char [] c = getSSN().toCharArray();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
         case '&':
          sb.append("&amp;");
          break;
         case '"':
          sb.append("&quot;");
          break;
```

```
                    case '\"':
                     sb.append("&quot;");
                     break;
                    default:
                     if (Character.isDefined(c[x]))
                      sb.append(c[x]);
                    }
                   }
                   return sb.toString();
                  }
                  public void setSSN(String inp){
                        this.aSSN = inp;
                  }
                  public void sSNFromXML(String n){
                        setSSN(n);
                  }
}

package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class PrototypeService extends Document {
          public static final String DOC_TYPE = "prototype.service";
           String mServiceName;
           String [] mServiceTerms;
           String [] mServiceLocation;
           ServiceOperation [] mServiceOperation;
          public PrototypeService(){
                super(DOC_TYPE);
         mServiceName = new String();
         mServiceTerms = new String[0];
         mServiceLocation = new String[0];
                this.mServiceOperation = new ServiceOperation[0];
         }
         public PrototypeService(String doc_type){
```

```
            super(doc_type);
        mServiceName = new String();
        mServiceTerms = new String[0];
        mServiceLocation = new String[0];
            this.mServiceOperation = new ServiceOperation[0];
        }
        static public PrototypeService initPrototypeService(String iServiceName,String []
iServiceTerms,String [] iServiceLocation,ServiceOperation [] iServiceOperation){
            PrototypeService obj = new PrototypeService();
            obj.initializeAll(iServiceName, iServiceTerms, iServiceLocation,
iServiceOperation);
            return obj;
        }

        public void initializeAll(String iServiceName,String [] iServiceTerms,String []
iServiceLocation,ServiceOperation [] iServiceOperation){
            mServiceName = iServiceName;
            mServiceTerms = iServiceTerms;
            mServiceLocation = iServiceLocation;
            mServiceOperation = iServiceOperation;
        }
        public String getServiceName(){
         return mServiceName;
        }
        public String getServiceNameToXML(){
            if (getServiceName() == null) return null;
            char [] c = getServiceName().toCharArray();
            StringBuffer sb = new StringBuffer();
            for (int x = 0; x < c.length; x++){
             switch(c[x]){
             case '>':
              sb.append("&gt;");
              break;
             case '<':
              sb.append("&lt;");
```

```
                break;
              case '&':
                sb.append("&amp;");
                break;
              case '"':
                sb.append("&quot;");
                break;
              case '\'':
                sb.append("&quot;");
                break;
              default:
                if (Character.isDefined(c[x]))
                  sb.append(c[x]);
              }
            }
            return sb.toString();
        }
        public void setServiceName(String inp){
            this.mServiceName = inp;
        }
        public void serviceNameFromXML(String n){
            setServiceName(n);
        }
        public String [] getServiceTerms(){
         return mServiceTerms;
        }
        public String getServiceTerms(int index){
            if (this.mServiceTerms == null)
              return null;
            if (index >= this.mServiceTerms.length)
              return null;
            if (index < 0 && -index > this.mServiceTerms.length)
              return null;
            if (index >= 0) return this.mServiceTerms[index];
            return this.mServiceTerms[this.mServiceTerms.length + index];
```

```
}
public String [] getServiceTermsToXML(){
        String [] valArr = getServiceTerms();
        if (valArr == null) return null;
        String [] nvArr = new String[valArr.length];
        for (int z = 0; z < nvArr.length; z++){
        char [] c = valArr[z].toCharArray();
        StringBuffer st = new StringBuffer();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
        case '&':
          sb.append("&amp;");
          break;
        case '"':
          sb.append("&quot;");
          break;
        case '':
          sb.append("&quot;");
          break;
        default:
          if (Character.isDefined(c[x]))
           sb.append(c[x]);
         }
        }
       nvArr[z] = sb.toString();
       }
       return nvArr;
}
```

```
    public void setServiceTerms(int index, String inp){
        if (this.mServiceTerms == null) {
            if (index < 0) {
             this.mServiceTerms = new String[1];
             this.mServiceTerms[0] = inp;
            } else {
             this.mServiceTerms = new String[index + 1];
             this.mServiceTerms[index] = inp;
            }
        } else if (index < 0) {
                    String [] newServiceTerms = new String[this.mServiceTerms.length
+ 1];
                    java.lang.System.arraycopy((Object)mServiceTerms, 0,
(Object)newServiceTerms, 0, this.mServiceTerms.length);
                    newServiceTerms[newServiceTerms.length - 1] = inp;
                    mServiceTerms = newServiceTerms;
            } else if (index >= this.mServiceTerms.length){
                    String [] newServiceTerms = new String[index + 1];
                    java.lang.System.arraycopy((Object)mServiceTerms, 0,
(Object)newServiceTerms, 0, this.mServiceTerms.length);
                    newServiceTerms[index] = inp;
                    mServiceTerms = newServiceTerms;
            } else {
                    this.mServiceTerms[index] = inp;
            }
    }
    public void setServiceTerms(String [] inp){
            this.mServiceTerms = inp;
    }
    public void serviceTermsFromXML(String n){
            setServiceTerms(-1, n);
    }
    public String [] getServiceLocation(){
     return mServiceLocation;
    }
```

```
public String getServiceLocation(int index){
        if (this.mServiceLocation == null)
          return null;
        if (index >= this.mServiceLocation.length)
          return null;
        if (index < 0 && -index > this.mServiceLocation.length)
          return null;
        if (index >= 0) return this.mServiceLocation[index];
        return this.mServiceLocation[this.mServiceLocation.length + index];
}
public String [] getServiceLocationToXML(){
        String [] valArr = getServiceLocation();
        if (valArr == null) return null;
        String [] nvArr = new String[valArr.length];
        for (int z = 0; z < nvArr.length; z++){
        char [] c = valArr[z].toCharArray();
        StringBuffer st = new StringBuffer();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
         case '&':
          sb.append("&amp;");
          break;
         case '"':
          sb.append("&quot;");
          break;
         case '':
          sb.append("&quot;");
          break;
```

```
                default:
                 if (Character.isDefined(c[x]))
                  sb.append(c[x]);
                }
               }
               nvArr[z] = sb.toString();
               }
               return nvArr;
          }
          public void setServiceLocation(int index, String inp){
             if (this.mServiceLocation == null) {
                  if (index < 0) {
                   this.mServiceLocation = new String[1];
                   this.mServiceLocation[0] = inp;
                  } else {
                   this.mServiceLocation = new String[index + 1];
                   this.mServiceLocation[index] = inp;
                  }
             } else if (index < 0) {
                        String [] newServiceLocation = new
String[this.mServiceLocation.length + 1];
                        java.lang.System.arraycopy((Object)mServiceLocation, 0,
(Object)newServiceLocation, 0, this.mServiceLocation.length);
                        newServiceLocation[newServiceLocation.length - 1] = inp;
                        mServiceLocation = newServiceLocation;
                  } else if (index >= this.mServiceLocation.length){
                        String [] newServiceLocation = new String[index + 1];
                        java.lang.System.arraycopy((Object)mServiceLocation, 0,
(Object)newServiceLocation, 0, this.mServiceLocation.length);
                        newServiceLocation[index] = inp;
                        mServiceLocation = newServiceLocation;
                  } else {
                        this.mServiceLocation[index] = inp;
                  }
          }
```

```
public void setServiceLocation(String [] inp){
        this.mServiceLocation = inp;
}
public void serviceLocationFromXML(String n){
        setServiceLocation(-1, n);
}
public ServiceOperation [] getServiceOperation(){
 return mServiceOperation;
}
public ServiceOperation getServiceOperation(int index){
      if (this.mServiceOperation == null)
       return null;
      if (index >= this.mServiceOperation.length)
       return null;
      if (index < 0 && -index > this.mServiceOperation.length)
       return null;
      if (index >= 0) return this.mServiceOperation[index];
      return this.mServiceOperation[this.mServiceOperation.length + index];
}
public void setServiceOperation(int index, ServiceOperation inp){
   if (this.mServiceOperation == null) {
      if (index < 0) {
       this.mServiceOperation = new ServiceOperation[1];
       this.mServiceOperation[0] = inp;
      } else {
       this.mServiceOperation = new ServiceOperation[index + 1];
       this.mServiceOperation[index] = inp;
      }
   } else if (index < 0) {
            ServiceOperation [] newServiceOperation = new
ServiceOperation[this.mServiceOperation.length + 1];
            java.lang.System.arraycopy((Object)mServiceOperation, 0,
(Object)newServiceOperation, 0, this.mServiceOperation.length);
            newServiceOperation[newServiceOperation.length - 1] = inp;
            mServiceOperation = newServiceOperation;
```

```
                } else if (index >= this.mServiceOperation.length){
                        ServiceOperation [] newServiceOperation = new
ServiceOperation[index + 1];
                        java.lang.System.arraycopy((Object)mServiceOperation, 0,
(Object)newServiceOperation, 0, this.mServiceOperation.length);
                        newServiceOperation[index] = inp;
                        mServiceOperation = newServiceOperation;
                } else {
                        this.mServiceOperation[index] = inp;
                }
        }
        public void setServiceOperation(ServiceOperation [] inp){
                this.mServiceOperation = inp;
        }
}

package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class Service extends Document {
        public static final String DOC_TYPE = "service";
        String mServiceName;
        String mServiceLocation;
        ServiceOperation [] mServiceOperation;
        String mServiceTerms;
        public Service(){
                super(DOC_TYPE);
        mServiceName = new String();
        mServiceLocation = new String();
                this.mServiceOperation = new ServiceOperation[0];
        mServiceTerms = new String();
        }
        public Service(String doc_type){
                super(doc_type);
        mServiceName = new String();
```

```
        mServiceLocation = new String();
                this.mServiceOperation = new ServiceOperation[0];
        mServiceTerms = new String();
        }
        static public Service initService(String iServiceName,String
iServiceLocation,ServiceOperation [] iServiceOperation,String iServiceTerms){
                Service obj = new Service();
                obj.initializeAll(iServiceName, iServiceLocation, iServiceOperation,
iServiceTerms);
                return obj;
        }

        public void initializeAll(String iServiceName,String
iServiceLocation,ServiceOperation [] iServiceOperation,String iServiceTerms){
                mServiceName = iServiceName;
                mServiceLocation = iServiceLocation;
                mServiceOperation = iServiceOperation;
                mServiceTerms = iServiceTerms;
        }
        public String getServiceName(){
         return mServiceName;
        }
        public String getServiceNameToXML(){
                if (getServiceName() == null) return null;
                char [] c = getServiceName().toCharArray();
                StringBuffer sb = new StringBuffer();
                for (int x = 0; x < c.length; x++){
                 switch(c[x]){
                 case '>':
                  sb.append("&gt;");
                  break;
                 case '<':
                  sb.append("&lt;");
                  break;
                 case '&':
```

```
                sb.append("&amp;");
                break;
              case '"':
                sb.append("&quot;");
                break;
              case '\'':
                sb.append("&quot;");
                break;
              default:
                if (Character.isDefined(c[x]))
                  sb.append(c[x]);
              }
            }
            return sb.toString();
    }
    public void setServiceName(String inp){
            this.mServiceName = inp;
    }
    public void serviceNameFromXML(String n){
            setServiceName(n);
    }
    public String getServiceLocation(){
     return mServiceLocation;
    }
    public String getServiceLocationToXML(){
            if (getServiceLocation() == null) return null;
            char [] c = getServiceLocation().toCharArray();
            StringBuffer sb = new StringBuffer();
            for (int x = 0; x < c.length; x++){
             switch(c[x]){
             case '>':
               sb.append("&gt;");
               break;
             case '<':
               sb.append("&lt;");
```

```
                break;
              case '&':
                sb.append("&amp;");
                break;
              case '"':
                sb.append("&quot;");
                break;
              case '\'':
                sb.append("&quot;");
                break;
              default:
                if (Character.isDefined(c[x]))
                  sb.append(c[x]);
              }
            }
            return sb.toString();
}
public void setServiceLocation(String inp){
        this.mServiceLocation = inp;
}
public void serviceLocationFromXML(String n){
        setServiceLocation(n);
}
public ServiceOperation [] getServiceOperation(){
 return mServiceOperation;
}
public ServiceOperation getServiceOperation(int index){
        if (this.mServiceOperation == null)
          return null;
        if (index >= this.mServiceOperation.length)
          return null;
        if (index < 0 && -index > this.mServiceOperation.length)
          return null;
        if (index >= 0) return this.mServiceOperation[index];
        return this.mServiceOperation[this.mServiceOperation.length + index];
```

```
    }
    public void setServiceOperation(int index, ServiceOperation inp){
        if (this.mServiceOperation == null) {
            if (index < 0) {
             this.mServiceOperation = new ServiceOperation[1];
             this.mServiceOperation[0] = inp;
            } else {
             this.mServiceOperation = new ServiceOperation[index + 1];
             this.mServiceOperation[index] = inp;
            }
        } else if (index < 0) {
                    ServiceOperation [] newServiceOperation = new
ServiceOperation[this.mServiceOperation.length + 1];
                    java.lang.System.arraycopy((Object)mServiceOperation, 0,
(Object)newServiceOperation, 0, this.mServiceOperation.length);
                    newServiceOperation[newServiceOperation.length - 1] = inp;
                    mServiceOperation = newServiceOperation;
            } else if (index >= this.mServiceOperation.length){
                    ServiceOperation [] newServiceOperation = new
ServiceOperation[index + 1];
                    java.lang.System.arraycopy((Object)mServiceOperation, 0,
(Object)newServiceOperation, 0, this.mServiceOperation.length);
                    newServiceOperation[index] = inp;
                    mServiceOperation = newServiceOperation;
            } else {
                    this.mServiceOperation[index] = inp;
            }
    }
    public void setServiceOperation(ServiceOperation [] inp){
            this.mServiceOperation = inp;
    }
    public String getServiceTerms(){
     return mServiceTerms;
    }
    public String getServiceTermsToXML(){
```

```
        if (getServiceTerms() == null) return null;
        char [] c = getServiceTerms().toCharArray();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
         case '&':
          sb.append("&amp;");
          break;
         case '"':
          sb.append("&quot;");
          break;
         case '\':
          sb.append("&quot;");
          break;
         default:
          if (Character.isDefined(c[x]))
           sb.append(c[x]);
         }
        }
        return sb.toString();
    }
    public void setServiceTerms(String inp){
        this.mServiceTerms = inp;
    }
    public void serviceTermsFromXML(String n){
        setServiceTerms(n);
    }
}
```

```
package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class ServiceOperation extends Document {
        public static final String DOC_TYPE = "service.operation";
         String mServiceOperationName;
         String mServiceOperationLocation;
         String mServiceOperationInput;
         String mServiceOperationOutput;
         public ServiceOperation(){
                 super(DOC_TYPE);
        mServiceOperationName = new String();
        mServiceOperationLocation = new String();
        mServiceOperationInput = new String();
        mServiceOperationOutput = new String();
        }
        public ServiceOperation(String doc_type){
                 super(doc_type);
        mServiceOperationName = new String();
        mServiceOperationLocation = new String();
        mServiceOperationInput = new String();
        mServiceOperationOutput = new String();
        }
        static public ServiceOperation initServiceOperation(String
iServiceOperationName,String iServiceOperationLocation.String iServiceOperationInput,String
iServiceOperationOutput){
                 ServiceOperation obj = new ServiceOperation();
                 obj.initializeAll(iServiceOperationName. iServiceOperationLocation,
iServiceOperationInput, iServiceOperationOutput);
                 return obj;
        }

        public void initializeAll(String iServiceOperationName,String
iServiceOperationLocation,String iServiceOperationInput,String iServiceOperationOutput){
                 mServiceOperationName = iServiceOperationName;
```

```
        mServiceOperationLocation = iServiceOperationLocation;
        mServiceOperationInput = iServiceOperationInput;
        mServiceOperationOutput = iServiceOperationOutput;
    }
    public String getServiceOperationName(){
     return mServiceOperationName;
    }
    public String getServiceOperationNameToXML(){
        if (getServiceOperationName() == null) return null;
        char [] c = getServiceOperationName().toCharArray();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
         case '&':
          sb.append("&amp;");
          break;
         case '"':
          sb.append("&quot;");
          break;
         case '\':
          sb.append("&quot;");
          break;
         default:
          if (Character.isDefined(c[x]))
           sb.append(c[x]);
        }
       }
       return sb.toString();
}
```

```
public void setServiceOperationName(String inp){
        this.mServiceOperationName = inp;
}
public void serviceOperationNameFromXML(String n){
        setServiceOperationName(n);
}
public String getServiceOperationLocation(){
 return mServiceOperationLocation;
}
public String getServiceOperationLocationToXML(){
        if (getServiceOperationLocation() == null) return null;
        char [] c = getServiceOperationLocation().toCharArray();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
        case '&':
          sb.append("&amp;");
          break;
        case '"':
          sb.append("&quot;");
          break;
        case '\'':
          sb.append("&quot;");
          break;
        default:
         if (Character.isDefined(c[x]))
          sb.append(c[x]);
        }
       }
```

```
        return sb.toString();
}
public void setServiceOperationLocation(String inp){
        this.mServiceOperationLocation = inp;
}
public void serviceOperationLocationFromXML(String n){
        setServiceOperationLocation(n);
}
public String getServiceOperationInput(){
 return mServiceOperationInput;
}
public String getServiceOperationInputToXML(){
        if (getServiceOperationInput() == null) return null;
        char [] c = getServiceOperationInput().toCharArray();
        StringBuffer sb = new StringBuffer();
        for (int x = 0; x < c.length; x++){
         switch(c[x]){
         case '>':
          sb.append("&gt;");
          break;
         case '<':
          sb.append("&lt;");
          break;
         case '&':
          sb.append("&amp;");
          break;
         case '"':
          sb.append("&quot;");
          break;
         case '\':
          sb.append("&quot;");
          break;
        default:
         if (Character.isDefined(c[x]))
          sb.append(c[x]);
```

```
            }
           }
            return sb.toString();
 }
 public void setServiceOperationInput(String inp){
            this.mServiceOperationInput = inp;
 }
 public void serviceOperationInputFromXML(String n){
            setServiceOperationInput(n);
 }
 public String getServiceOperationOutput(){
  return mServiceOperationOutput;
 }
 public String getServiceOperationOutputToXML(){
           if (getServiceOperationOutput() == null) return null;
           char [] c = getServiceOperationOutput().toCharArray();
           StringBuffer sb = new StringBuffer();
           for (int x = 0; x < c.length; x++){
            switch(c[x]){
            case '>':
             sb.append("&gt;");
             break;
            case '<':
             sb.append("&lt;");
             break;
           case '&':
            sb.append("&amp;");
            break;
          case '"':
           sb.append("&quot;");
           break;
         case '\':
          sb.append("&quot;");
          break;
        default:
```

```
                    if (Character.isDefined(c[x]))
                      sb.append(c[x]);
                    }
                  }
                  return sb.toString();
          }
          public void setServiceOperationOutput(String inp){
                  this.mServiceOperationOutput = inp;
          }
          public void serviceOperationOutputFromXML(String n){
                  setServiceOperationOutput(n);
          }
}

package com.veo.xdk.dev.schema.test.blib;

import com.veo.vsp.doclet.meta.Document;
public class ServiceSet extends Document {
        public static final String DOC_TYPE = "service.set";
         Service [] mService;
        public ServiceSet(){
                super(DOC_TYPE);
                this.mService = new Service[0];
        }
        public ServiceSet(String doc_type){
                super(doc_type);
                this.mService = new Service[0];
        }
        static public ServiceSet initServiceSet(Service [] iService){
                ServiceSet obj = new ServiceSet();
                obj.initializeAll(iService);
                return obj;
        }

        public void initializeAll(Service [] iService){
```

```
            mService = iService;
        }
        public Service [] getService(){
         return mService;
        }
        public Service getService(int index){
            if (this.mService == null)
             return null;
            if (index >= this.mService.length)
              return null;
            if (index < 0 && -index > this.mService.length)
              return null;
            if (index >= 0) return this.mService[index];
            return this.mService[this.mService.length + index];
        }
        public void setService(int index, Service inp){
          if (this.mService == null) {
              if (index < 0) {
               this.mService = new Service[1];
               this.mService[0] = inp;
              } else {
               this.mService = new Service[index + 1];
               this.mService[index] = inp;
              }
          } else if (index < 0) {
                    Service [] newService = new Service[this.mService.length + 1];
                    java.lang.System.arraycopy((Object)mService, 0,
(Object)newService, 0, this.mService.length);
                    newService[newService.length - 1] = inp;
                    mService = newService;
              } else if (index >= this.mService.length){
                    Service [] newService = new Service[index + 1];
                    java.lang.System.arraycopy((Object)mService, 0,
(Object)newService, 0, this.mService.length);
                    newService[index] = inp;
```

```
                mService = newService;
            } else {
                this.mService[index] = inp;
            }
        }
        public void setService(Service [] inp){
            this.mService = inp;
        }
    }
```

上記したＪＡＶＡビーンに加えて、変換コードは以下に示されているように、ＪＡＶＡからＸＭＬへ、及びＸＭＬからＪＡＶＡへ翻訳するため作られる。

Java to XML

```
<!DOCTYPE tree SYSTEM "tree.dtd">
<tree source = "null" pass-through = "false">
<before>
 <vardef name = "attribute.def">
  <element source = "ATTRIBUTE" class = "NAME" type = "5" position = "-2">
   <parse>
    <data class = "java.lang.String" position = "-2"/>
   </parse>
  </element>
 </vardef>
 <verdef name = "pcdata.def">
  <element source = "PCDATA" class = "NAME" type = "4" position = "-2">
   <parse>
    <data class = "999" type = "6" position = "-2"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "content.def">
  <element source = "PCDATA">
```

```
        <parse>
         <data class = "999" type = "6" position = "-2"/>
        </parse>
       </element>
      </vardef>
      <vardef name = "ServiceSet.var">
      <element source = "com.veo.xdk.dev.schema.test.blib.ServiceSet" class = "service.set" type =
     "4" position = "-2">
       <parse>
        <callvar name = "Service.var"/>
       </parse>
      </element>
      </vardef>
      <vardef name = "PrototypeService.var">
      <element source = "com.veo.xdk.dev.schema.test.blib.PrototypeService" class =
    "prototype.service" type = "4" position = "-2">
       <parse>
        <callvar name = "pcdata.def" parms = "setSource ServiceNameToXML setGenerator
    service.name"/>
        <callvar name = "pcdata.def" parms = "setSource ServiceTermsToXML setGenerator
    service.terms"/>
        <callvar name = "pcdata.def" parms = "setSource ServiceLocationToXML setGenerator
    service.location"/>
        <callvar name = "ServiceOperation.var"/>
       </parse>
      </element>
      </vardef>
      <vardef name = "Service.var">
      <element source = "com.veo.xdk.dev.schema.test.blib.Service" class = "service" type = "8"
    position = "0">
       <parse>
        <callvar name = "pcdata.def" parms = "setSource ServiceNameToXML setGenerator
    service.name"/>
        <callvar name = "pcdata.def" parms = "setSource ServiceLocationToXML setGenerator
    service.location"/>
```

```
      <callvar name = "ServiceOperation.var"/>
      <callvar name = "pcdata.def" parms = "setSource ServiceTermsToXML setGenerator
  service.terms"/>
    </parse>
   </element>
   </vardef>
   <vardef name = "ServiceOperation.var">
   <element source = "com.veo.xdk.dev.schema.test.blib.ServiceOperation" class =
  "service.operation" type = "4" position = "-2">
    <parse>
     <callvar name = "pcdata.def" parms = "setSource ServiceOperationNameToXML
  setGenerator service.operation.name"/>
     <callvar name = "pcdata.def" parms = "setSource ServiceOperationLocationToXML
  setGenerator service.operation.location"/>
     <callvar name = "pcdata.def" parms = "setSource ServiceOperationInputToXML
  setGenerator service.operation.input"/>
     <callvar name = "pcdata.def" parms = "setSource ServiceOperationOutputToXML
  setGenerator service.operation.output"/>
    </parse>
   </element>
   </vardef>
  </before>
  <parse>
  <callvar name = "ServiceSet.var"/>
  <callvar name = "PrototypeService.var"/>
  <callvar name = "Service.var"/>
  <callvar name = "ServiceOperation.var"/>
  </parse>
  </tree>

  XML to Java

  <!DOCTYPE tree SYSTEM "tree.dtd">
  <tree source = "null" pass-through = "false">
  <before>
```

```
<vardef name = "business.var">
 <element source = "business"
      class = "com.veo.xdk.dev.schema.test.blib.Business"
      type = "7" setter = "setBusiness">
  <before>
   <onattribute name = "business.number">
    <actions>
     <callmeth name = "businessNumberFromXML">
      <parms>
       <getattr name = "business.number"/>
      </parms>
     </callmeth>
    </actions>
   </onattribute>
  </before>
  <parse>
   <callvar name = "party.name.var" parms = "setPosition -1"/>
   <callvar name = "address.set.var"/>
  </parse>
 </element>
</vardef>
<vardef name = "party.name.var">
 <element source = "party.name" setter = "partyNameFromXML" position = "-1" class =
"java.lang.String">
  <parse>
   <data class = "java.lang.String" position = "0"/>
  </parse>
 </element>
</vardef>
<vardef name = "city.var">
 <element source = "city" setter = "cityFromXML" position = "-1" class = "java.lang.String">
  <parse>
   <data class = "java.lang.String" position = "0"/>
  </parse>
 </element>
```

```
 </vardef>
 <vardef name = "internet.var">
  <element source = "internet" setter = "internetFromXML" position = "-1" class =
"java.lang.String">
   <parse>
    <data class = "java.lang.String" position = "0"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "country.var">
  <element source = "country" setter = "countryFromXML" position = "-1" class =
"java.lang.String">
   <parse>
    <data class = "java.lang.String" position = "0"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "state.var">
  <element source = "state" setter = "stateFromXML" position = "-1" class = "java.lang.String">
   <parse>
    <data class = "java.lang.String" position = "0"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "email.var">
  <element source = "email" setter = "emailFromXML" position = "-1" class =
"java.lang.String">
   <parse>
    <data class = "java.lang.String" position = "0"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "address.physical.var">
  <element source = "address.physical"
      class = "com.veo.xdk.dev.schema.test.blib.AddressPhysical"
```

```
        type = "7" setter = "setAddressPhysical">
      <before>
      </before>
      <parse>
       <callvar name = "street.var" parms = "setPosition -1"/>
       <callvar name = "city.var" parms = "setPosition -1"/>
       <callvar name = "state.var" parms = "setPosition -1"/>
       <callvar name = "postcode.var" parms = "setPosition -1"/>
       <callvar name = "country.var" parms = "setPosition -1"/>
      </parse>
    </element>
  </vardef>
  <vardef name = "telephone.var">
   <element source = "telephone" setter = "telephoneFromXML" position = "-1" class =
"java.lang.String">
    <parse>
     <data class = "java.lang.String" position = "0"/>
    </parse>
   </element>
  </vardef>
  <vardef name = "person.var">
   <element source = "person"
        class = "com.veo.xdk.dev.schema.test.blib.Person"
        type = "7" setter = "setPerson">
    <before>
     <onattribute name = "SSN">
      <actions>
       <callmeth name = "sSNFromXML">
        <parms>
         <getattr name = "SSN"/>
        </parms>
       </callmeth>
      </actions>
     </onattribute>
    </before>
```

```
    <parse>
     <callvar name = "party.name.var" parms = "setPosition -1"/>
     <callvar name = "address.set.var"/>
    </parse>
   </element>
  </vardef>
  <vardef name = "fax.var">
   <element source = "fax" setter = "faxFromXML" position = "-1" class = "java.lang.String">
    <parse>
     <data class = "java.lang.String" position = "0"/>
    </parse>
   </element>
  </vardef>
  <vardef name = "street.var">
   <element source = "street" setter = "streetFromXML" position = "-1" class =
  "java.lang.String">
    <parse>
     <data class = "java.lang.String" position = "0"/>
    </parse>
   </element>
  </vardef>
  <vardef name = "address.set.var">
   <element source = "address.set"
        class = "com.veo.xdk.dev.schema.test.blib.AddressSet"
        type = "7" setter = "setAddressSet">
    <before>
    </before>
    <parse>
     <callvar name = "address.physical.var"/>
     <callvar name = "telephone.var" parms = "setPosition -1"/>
     <callvar name = "fax.var" parms = "setPosition -1"/>
     <callvar name = "email.var" parms = "setPosition -1"/>
     <callvar name = "internet.var" parms = "setPosition -1"/>
    </parse>
   </element>
```

```
 </vardef>
 <vardef name = "postcode.var">
  <element source = "postcode" setter = "postcodeFromXML" position = "-1" class =
 "java.lang.String">
   <parse>
     <data class = "java.lang.String" position = "0"/>
   </parse>
  </element>
 </vardef>
 <vardef name = "market.participant.var">
  <element source = "market.participant"
        class = "com.veo.xdk.dev.schema.test.blib.MarketParticipant"
        type = "7" position = "0">
   <before>
   </before>
   <parse>
    <callvar name = "business.var"/>
    <callvar name = "person.var"/>
   </parse>
  </element>
 </vardef>
 </before>
 <parse>
 <callvar name = "business.var"/>
 <callvar name = "party.name.var"/>
 <callvar name = "city.var"/>
 <callvar name = "internet.var"/>
 <callvar name = "country.var"/>
 <callvar name = "state.var"/>
 <callvar name = "email.var"/>
 <callvar name = "address.physical.var"/>
 <callvar name = "telephone.var"/>
 <callvar name = "person.var"/>
 <callvar name = "fax.var"/>
 <callvar name = "street.var"/>
```

```
<callvar name = "address.set.var"/>
<callvar name = "postcode.var"/>
<callvar name = "market.participant.var"/>
</parse>
</tree>
```

Makefiles:

```
#
# this makefile was generated by bic version 0.0. 05/02/1998
#
#
#
#

# get the package name from the package argument passed to SchemaGen
PACKAGE_NAME = com/veo/xdk/dev/schema/test/blib

JAVA_SOURCES    +=      \
        MarketParticipant.java \
        Business.java \
        Person.java \
        Party.java \
        AddressPhysical.java \
        AddressSet.java \

MAKEFILE_MASTER_DIR = xxx
include $(MAKEFILE_MASTER_DIR)/Makefile.master

all:: $(JAVA_CLASSES)

#
# this makefile was generated by bic version 0.0. 05/02/1998
#
```

```
#
#
#

# get the package name from the package argument passed to SchemaGen
PACKAGE_NAME = com/veo/xdk/dev/schema/test/blib

JAVA_SOURCES    +=          \
        ServiceSet.java \
        PrototypeService.java \
        Service.java \
        ServiceOperation.java \

MAKEFILE_MASTER_DIR = xxx
include $(MAKEFILE_MASTER_DIR)/Makefile.master

all:: $(JAVA_CLASSES)
```

Finally, the XML document instances generated at run time according to the model above for one example follows:

```
<!DOCTYPE market.participant SYSTEM "market.participant.dtd" >
<market.participant>

<business business.number="1234567890" >

<party.name>IBM</party.name>

<address.set>
<address.physical>

<street>1 IBM Way</street>
<city>Palo Alto</city>
<state>CA</state>
```

<postcode>94304</postcode>
<country>USA</country>
</address.physical>

<telephone>123 456-7890</telephone>
<fax>123 456 0987</fax>
<email>ibmec@ibm.com</email>
</address.set>

</business>

</market.participant>

<!DOCTYPE service SYSTEM "service.dtd" >
<service.set>
<service>
<service.name>Order Service</service.name>
<service.location>www.ibm.com/order</service.location>

<service.operation>
<service.operation.name>Submit Order</service.operation.name>
<service.operation.location>www.ibm.com/order/submit</service.location>
<service.operation.input>um:x-ibm:services:order:operations:po.dtd</service.operation.input>
<service.operation.output>um:x-
ibm:services:order:operations:poack.dtd</service.operation.output>
</service.operation>

<service.operation>
<service.operation.name>Track Order</service.operation.name>
<service.operation.location>www.ibm.com/order/track</service.location>
<service.operation.input>um:x-
ibm:services:order:operations:track.irequest.dtd</service.operation.input>
<service.operation.output>um:x-
ibm:services:order:operations:track.iresponse.dtd</service.operation.output>
</service.operation>

</service>
(/service.set>

【０１０１】

ドラッグ、ドロップ及び形を編集するユーサインターフェースを提供する、ＢＩＤコンポーザ（ｃｏｍｐｏｓｅｒ）アプリケーションと一緒にツールを使用して、開発者は、ビジネスインターフェースの定義を作り出すことができ、及び、良好に形成された、有効なビジネスインターフェースの定義をＸＭＬドキュメントの形態で生産することができる。このように、例のランタイム・インスタンスは、ＩＢＭからラップトップコンピュータを注文するために、イングラム・マイクロ（Ｉｎｇｒａｍ　Ｍｉｃｒｏ）によって使用される、ＩＢＭのためのサービスを注文するためのビジネスインターフェースの定義である。（出願人と、ＩＢＭ又はイングラム・マイクロの間には関係はない。）これらのプロセスを利用すると、ユーザは、本発明に従って定義されたドキュメントを使用して、ビジネスインターフェースのプログラムができるシステムを構築することができる。

【０１０２】

ＸＭＬ／ＪＡＶＡ環境における本発明のＣＢＬ及びＢＩＤプロセッサの役割は購入注文の処理の以下の説明によりさらに理解できるであろう。

【０１０３】

会社ＡはＣＢＬ　ＤＴＤ及びモジュールのライブラリを含む視覚プログラム環境を使用するその購入注文ドキュメントを定義し、全ては共通のビジネス言語エレメントを使用して定義され、それらがデータタイプ及び他の翻訳情報を含むようになっている。会社ＡのＰＯはＣＢＬライブラリに付属しているより一般的なトランザクションドキュメント仕様に対してマイナなカスタマイゼイションを含むかもしれないし、或いは、アドレス、データ及び時間、電流等のためＣＢＬモジュールからのグランドから作られるかもしれない。

【０１０４】

（上記したｔｒａｎｓａｃｔ．ｄｔｄのような）一般的なトランザクションドキュメント仕様のドキュメントはＣＢＬ仕様がモジュールから作られ、他のＣＢＬ　ＤＴＤと連結される方法を類型化する。

【０１０５】

コンパイラは購入注文の定義を取り、幾つかの異なるターゲット形式を発生する。これらのターゲット形式のすべては元の仕様の「ツリーツーツリー（ｔｒｅ

ｅ　ｔｏ　ｔｒｅｅ）」変形を介して得ることができる。この例にとって最も重要なのは、

（ａ）購入注文のためのXML　ＤＴＤ

（ｂ）購入注文のためのデータ構造を含むＪＡＶＡビーン（Ｂｅａｎ）（ＪＡＶＡクラス、引数、データタイプ、方法、及び例外構造は購入注文のスキーマ定義の情報に対応して作り出される。）

（ｃ）購入注文ＤＴＤに従う購入注文を購入注文ＪＡＶＡビーンに変換し、又は、それらをデータベースにロードし、又は、ブラウザに購入注文を表示するためＨＴＭＬ（又はＸＳＬスタイル集）を作り出す整列プログラム

（ｄ）購入注文ＪＡＶＡビーンからデータ値を引き出し、それらを購入注文ＤＴＤに従うXMLドキュメントに変換する整列しないプログラム

今、シナリオに戻ると、購入の申し込みは購入注文を発生し、該購入注文は購入注文を受け入れる供給者のためのサービスインターフェースとして明記されたＤＴＤに従う。

【０１０６】

パーサは購入注文ＤＴＤを使用し、エレメント及びそれが含む属性値についての情報ストリームへの購入注文インスタンスを分解する。その後、これらのプロパティセットはそれらをＪＡＶＡコードと重ねることにより対応するＪＡＶＡイベントオブジェクトに変換される。事実上、この変換は文法がＤＴＤにより定義される顧客プログラム言語の命令として印を付けたXMLドキュメントの部分を処理する。これらのＪＡＶＡイベントはコンパイラにより生成された整列しているアプリケーションにより処理され、ＪＡＶＡビーンのデータ構造をロードすることができる。

【０１０７】

XMLドキュメントをＪＡＶＡアプリケーションのための１セットのイベントに変換し、処理することは、解析をする正常なモデルとは異なり、解析が完了するまで内部データ構造及び処理は開始しないので、パーサ出力は維持される。ＢＩＤ定義に応じたイベントベースの処理は、最初のイベントが発せられると直ぐに同時に発生のドキュメントアプリケーション処理を始めさせるので、、プロセ

ッサの非常に優れた機能を可能にする鍵である。

【０１０８】

各種タイプのイベントに聞くＪＡＶＡプログラムはそれらのイベントのスキーマ定義から形成される。これらのリスナーはＣＢＬのＸＭＬ定義と関連するビジネス論理を実行するために作り出されたプログラムであり、例えば、アドレスエレメントと関連し、データベースをチェックすることにより郵便番号を確認するコードであってもよい。これらのリスナーはドキュメントルータで登録することによりイベントに応募し、それらに興味を示したすべての応募者に関連したイベントを割り当てる。

【０１０９】

新規のリスナーがプログラムするこの公表し、応募したアーキテクチャ手段は現存のものの知識なしに、またはそれに影響を与えることなく付加されることができる。各リスナーはルータがそのイベントを割り当てるキューを有し、複数のリスナーはそれら自身のペースで並行してイベントを取扱うことができる。

【０１１０】

ここの例示の購入注文にとって、

・注文入力プログラムにそれをつなげるであろう購入注文

・在庫をチェックするであろう製品説明書

・Ｆｅｄ　Ｅｘ又は引き渡しの有効性のための他のサービスをチェックすることができるであろうアドレス情報

・（信用貸しの価値のため、又は公告を提供するため、又は顧客が誰であるかを知ることに基づく同様の処理のため）注文の履歴をチェックすることができるであろう買い手の情報

のためのリスナーであるかもしれない。

【０１１１】

複合したリスナーは初期のものの形状として作り出されることができる。（例えば、購入注文のリスナーがここのこれらのリスナーを含み、引き合いに出してもよく、又は、それらはそれら自身で引き合いに出されてもよい。）

【０１１２】

図１１は図１のネットワークでのマーケットマーカのノードを示している。マーケットマーカのノードは図３のシステムの基本構造を含み、ネットワークインターフェース１１０１、ドキュメントパーサ１１０２、ホストへのドキュメント及びドキュメント翻訳機へのホスト１１０３、フロントエンド１１０４を有し、この例ではルータと呼ばれている。この例でのマーケットマーカモジュール１１０５は１セットのビジネスインターフェース定義、又はマーケットの参加者のためマーケットマーカの機能を支援するのに十分な他の識別子、ＣＢＬリポジトリ、及びマーケットの参加者の要求をすべて満たすコンパイラを備えている。ルータ１１０４は参加者のレジストリ及びドキュメントフィルタを備え、翻訳機の出力でパーサにより生成されたイベントに応答し、参加者のレジストリにより、及びＸＭＬイベント発生器に対するリスナーの間のエレメント及び属性フィルタにより、入力ドキュメントの経路を定める。したがって、マーケットの一定の参加者は登録し、予め特定したパラメータに合うドキュメントを受け取ってもよい。例えば、特定のＤＴＤによる入力ドキュメントは、閾値以上の購入製品数、又は購入されるドキュメント要求の最大価格等の属性を含み、ルータ１１０４でフィルタドキュメントに使用されることができる。その後、ルータ１１０４で参加者のレジストリに登録された情報に合致するようなドキュメントは登録された参加者に送られる。

【０１１３】

ルータ１１０４はまたローカルホストサービス１１０５及び１１０６の要求を満たし、例えば、マーケットマーカと同様にマーケットの参加者のように作動する。通常、ルータ１１０４により受け取られたドキュメントはトラバースされ、宛先を決定し、そのようなドキュメントは送られ、そこで再度、翻訳機１１０３を介して、必要であれば、ネットワークインターフェース１１０１からそれぞれの宛先に送られる。

【０１１４】

マーケットマーカは１セットの内部及び外部のビジネスサービスと一緒に結び付けるサーバであり、仮想の計画又はトレーディングコミュニティを作り出す。サーバは入力ドキュメントを解析し、例えば、製品データのためのリクエストを

カタログサーバに渡すことにより、又は、購入注文をＥＲＰシステムに送ることにより、適当なサービスを引き合いに出す。サーバはまた翻訳タスクを取扱い、トレーディングパートナにより使用されるドキュメント書式及びそのレガシーシステムにより要求されるデータ書式に会社のＸＭＬドキュメントからの情報をマッピングする。

【０１１５】

上記サービスの定義に関して、会社が購入注文を提示したとき、サーバのＸＭＬパーサは購入注文ＤＴＤを使用し、購入注文インスタンスを情報イベントのストリームに変換する。その後、これらのイベントは所定のタイプのイベントを取扱うためにプログラムされたアプリケーションに送られ、幾つかの場合には、情報はインターネットを介して異なるビジネスに完全に送られる。購入注文の例では、幾つかのアプリケーションはパーサから来る情報で作動してもよい。

・注文入力プログラムは完全なメッセージとして購入を処理する。

・ＥＲＰシステムは購入注文に記述された製品のための在庫をチェックする。

・顧客データベースは顧客のアドレスを証明し、又は更新する。

・運送会社はアドレス情報を使用し、搬送をスケジュールする。

・銀行はクレジットカードを使用し、トランザクションを許可する。

【０１１６】

トレーディングパートナーは、彼らが交換するビジネスドキュメントの構造、内容及び順序について同意するだけでよく、ＡＰＩの詳細について同意する必要はない。ドキュメントがどのように処理されるか及びどのようなアクション結果が生じるかは、厳密にサービスを提供するビジネス次第である。これは、システムレベルからビジネスレベルへの統合を高める。それは、ビジネスが、その内部技術の履行、組織化、又はプロセスにおける変更にもかかわらず、明瞭で安定なインターフェイスをそのビジネスパートナーに提供することを可能にする。

【０１１７】

図１２、１３及び１４は、図１１のシステムのマーケットメーカーノードにおいて実行されるプロセスを示している。図１２において、入力ドキュメントが、発信参加者ノードからのネットワークインターフェイスにおいて受け取られる（

ステップ１２００）。ドキュメントがパースされる（ステップ１２０１）。ドキュメントが、ホストのフォーマットへ、例えば、ＸＭＬからＪＡＶＡへ変換される（ステップ１２０２）。ホストフォーマット化されたイベント及びオブジェクトが、その後、ルータサービスへ送られる（ステップ１２０３）。ドキュメントのタイプ及びドキュメントの内容に従ってドキュメントを受け入れるように記憶されたサービスが識別される（ステップ１２０４）。ドキュメント及びドキュメントの一部が、識別されたサービスに送られる（ステップ１２０５）。ドキュメントの内容に応じてサービスが実行される（ステップ１２０６）。サービスの出力データが発生される（ステップ１２０７）。出力が、ドキュメントフォーマットへ、例えば、ＪＡＶＡフォーマットからＸＭＬフォーマットへ変換される（ステップ１２０８）。最後に、出力ドキュメントが参加者ノードへ送られる（ステップ１２０９）。

【０１１８】

レジストレーションサービスは、ルータにより管理されるそのような機能の一つである。従って、マーケット参加者ドキュメントが図１３に示されるネットワークインターフェイスにおいて受け入れられる（ステップ１３００）。マーケット参加者ドキュメントがマーケットメーカーノードについてのビジネスインターフェイス定義リポジトリに記憶される（ステップ１３０１）。加えて、ドキュメントがパースされる（ステップ１３０２）。パースされたドキュメントが、ホストのフォーマットへ変換される（ステップ１３０３）。次に、ドキュメントがルータサービスへ送られる（ステップ１３０４）。ルータサービスは、レジストレーションサービスをドキュメントのタイプ及びドキュメントの内容に従ってドキュメントの宛先であると識別するリスナーを含む（ステップ１３０５）。ドキュメント及びドキュメントのエレメントが、レジストレーションサービスに送られる（ステップ１３０６）。レジストレーションサービスにおいて、必要とされるサービス仕様がビジネスインターフェイス定義に応じてリトリーブされる（ステップ１３０７）。ステップ１３０８において、サービス仕様が集められる場合には、ビジネスインターフェイス定義及びサービス仕様に従ってルータサービスフィルタがセットされる（ステップ１３０９）。レジストレーションアクノリッジ

メントデータが発生される（ステップ１３１０）。レジストレーションアクノリッジメントデータが、ドキュメントフォーマットへ変換される（ステップ１３１１）。最後に、アクノリッジメントドキュメントが参加者ノードへ送られてマーケットメーカーで成功裏にレジスタされた参加者に通知する（ステップ１３１２）。

【０１１９】

必要とされるサービス仕様を集めるステップ１３０７におけるプロセスは、図１４の１つの例について示されている。このプロセスは、マーケット参加者によりサポートされるサービスビジネスインターフェイス定義を探し出すことにより始まる（ステップ１４００）。サービス定義が例えば、リポジトリノードへのＥメールトランザクション又はウェブアクセスによりリトリーブされる（ステップ１４０１）。サービス仕様がＢＩＤリポジトリに記憶される（ステップ１４０２）。サービスビジネスインターフェイス定義ドキュメントがパースされる（ステップ１４０３）。パースされたドキュメントが、ホストのフォーマットへ変換される（ステップ１４０４）。次に、ホストのオブジェクトがルータサービスへ送られる（ステップ１４０５）。レジストレーションサービスがドキュメントのタイプ及び内容に従って識別される（ステップ１４０６）。最後に、サービスビジネスインターフェイス定義ドキュメントの情報がレジストレーションサービスに送られて（ステップ１４０７）、図１３のプロセスに従って使用される。

【０１２０】

図１５は、本発明に従ってマーケットメーカーノードに入力するデータを処理するプロセッサ、コンポーネント及びシーケンスを示している。マーケットメーカーノードは、ネットワークインターフェイスにおいて通信エージェント１５００を有する。通信エージェントはＸＭＬパーサ１５０１と接続され、ＸＭＬパーサ１５０１は、イベントをＸＭＬプロセッサ１５０２に供給する。ＸＭＬプロセッサ１５０２は、イベントをドキュメントルータに供給する。ドキュメントルータは、ドキュメントサービス１５０４にフィードし、ドキュメントサービス１５０４は、受け取ったドキュメントをホストシステムのエンタープライズソルーションソフトウェア１５０５に供給するインターフェイスとなる。通信エージェン

ト１５００は、ＨＴＴＰ、ＳＭＴＰ、ＦＴＰ又は他のプロトコルのようなプロトコルを支持する適当なプロトコルスタックを有するインターネットインターフェイスである。従って、入力するデータは、特定の通信チャンネルに適合するようにＸＭＬ構文法、ＡＳＣＩＩデータ構文法又は他の構文法で入力してもよい。非ＸＭＬ構文法で受け取ったドキュメントは全てＸＭＬに変換されてＸＭＬパーサに送られる。非ＸＭＬ形式からＸＭＬ形式への変換をサポートするために変換テーブル１５０６が使用される。

【０１２１】

変換されたドキュメントはパーサ１５０１に供給される。ＸＭＬパーサは、受け取ったＸＭＬドキュメントを、それとマッチするドキュメントタイプ定義に従ってパースする。エラーが見つかった場合には、パーサがドキュメントを通信エージェント１５００へ送り返す。ビジネスインターフェイス定義コンパイラＢＩＤＣ１５０７は、ビジネスインターフェイス定義データのためのコンパイラとして働く。ＸＭＬパーサのためのＤＴＤファイル、ＤＴＤファイルに対応するＪＡＶＡビーンズ及びＤＴＤファイルをＪＡＶＡビーンズへ変換する変換ルールは、ＢＩＤデータをコンパイルすることにより作られる。これらのツールを参照することにより、ＸＭＬインスタンスがＪＡＶＡインスタンスへ変換される。そこで、ＢＩＤコンパイラ１５０７は、ＤＴＤドキュメント１５０８を記憶して１５０９に対応するＪＡＶＡドキュメントを発生する。ＸＭＬドキュメントはプロセッサ１５０２に送られ、プロセッサ１５０２は、そのドキュメントをＪＡＶＡフォーマットに変換する。好ましいシステムでは、ＸＭＬフォーマットで受け取ったドキュメントタイプ定義と同じステータスを有するＪＡＶＡドキュメントが発生される。ＪＡＶＡビーンズがドキュメントルータ１５０３に送られる。ドキュメントルータ１５０３がＪＡＶＡビーンズを受け取り、受け取ったクラスをレジストリプログラムを使用して、例えば、上述したイベントリスナーアーキテクチャを使用して、適当なドキュメントサービスに送る。ルータ１５０３からＪＡＶＡビーンズの形式でドキュメントを受け取ったドキュメントサービス１５０４は、エンタプライズソルーションソフトウェアとして働く。これは、ＸＭＬイベントのリスナーを入力するデータ流れと結合するレジストリサービス１５１０と、入

力するドキュメントの適当なサービスへのルーティングを管理するサービスマネージャ１５１１とを有する。ドキュメントサービスマネージャ１５１１は、レジストリサービスの管理及びドキュメントの一貫性の維持等を行う。

【０１２２】

ドキュメントサービスは、任意のプロプラエタリＡＰＩを使用して、又は、ＣＯＲＢＡ／ＣＯＭインターフェイスその他のアーキテクチャのようなより共通な形式を使用してバックエンドシステムと通信する。

【０１２３】

図１６は、本発明に係るマーケットメーカー及びマーケット参加者構造のヒューリスティックダイアグラムである。従って、本発明に係るＥコマーストランザクションマーケットは、図１６に示されているように論理的に組織化できる。その組織の最上部において、マーケットメーカーノード１６００が設定される。マーケットメーカーノードは、マーケットプレイス１６０１を設定するリソースを含む。このようなリソースは、マーケットレジストリサービス等を含む。ビジネス１６０２は、ビジネスインターフェイス定義をパブリッシュすることによりマーケットプレイスにレジスタする。ビジネスインターフェイス定義は、ビジネスが参加するコマーシャルトランザクションについてサービス１６０３を定義する。トランザクション１６０４及びサービス１６０３は、ドキュメント１６０５を使用して入力及び出力を定め、トランザクションにおける参加者の間の商業上の関係をアウトラインする。ドキュメントは各トランザクションの詳細を含むコンテント１６０６を有する。コンテントがマーケットの参加者により及びマーケットメーカーにより処理される方法は、本発明に従って設立されるＥコマーストランザクションネットワークに基づくドキュメントに完全に依存する。全体として、通信ネットワーク上でＥコマーストランザクション可能とするように頑強性、拡張性及び直観力に富む構造が提供される。

【０１２４】

従って、本発明は、例示のシステムにおいて、ＸＭＬプロセッサに基づくプラットフォームを提供し、疎結合されたビジネスシステムの間のインターフェイスとしてＸＭＬドキュメントを使用している。ドキュメントは、ビジネスの間で転

送され、会社のビジネスシステムに入る前に特定のノードによって処理される。従って、プラットフォームは、各ビジネスシステムが異なる内部コマースプラットフォーム、処理及び意味論を使用している場合に、共通の組のビジネスドキュメント及び形式を特定することによりビジネス間のＥコマースアプリケーションを可能にする。

【０１２５】

本発明によれば、ビジネスシステムとサービスを相互接続することによりバーチャルエンタプライズが形成され、主に、ビジネスが受け入れて発生するドキュメント（ＸＭＬコード化された）に間して次のとおり定義される。

「貴社が当社にカタログを請求される場合には、貴社にカタログを送ります。」

「貴社が購入の注文をされ当社がそれを受け入れことができる場合には、貴社に送り状を送ります。」

上述した本発明の好ましい実施例の説明は、本発明の解説するためになされたものである。これは、包括的であること又は発明を開示した厳密な形式に限定することを意図するものではない。当業者にとって多くの修正及び変更は明らかである。本発明の範囲は、以下の請求の範囲及びそれと等価なものによって定められるものである。

【図面の簡単な説明】

【図１】

本発明によるビジネスインタフェース定義（ＢＩＤ）を含む電子商取引ネットワークの概略図である。

【図２】

本発明によるビジネスインタフェース定義ドキュメントの概略図である。

【図３】

本発明によるネットワーク中の参加者ノードのためのサーバの概念的なブロックダイアグラムである。

【図４】

本発明による参加者ノードで受け取られたドキュメントのプロセシングを説明するフローチャートである。

【図５】

ＸＭＬベースのシステムについての構文解析プログラム及びトランザクションプロセスフロントエンドのブロックダイアグラムである。

【図６】

構文解析プログラム機能のフローの概念的なダイアグラムである。

【図７】

本発明によるビジネスインタフェース定義に使用されるサーバにおけるリソースの概略図である。

【図８】

ビジネスインタフェース定義の構築のために使用する、本発明によるリポジトリの概略図である。

【図９】

本発明によるビジネスインタフェース定義を構築するプロセスを説明するフローチャートである。

【図１０】

本発明によるリポジトリの発見的概要を提供するものである。

【図１１】

ビジネスインタフェース定義に基づく本発明のネットワークのためのマーケットメーカー機能を提供するサーバにおけるリソースの概略図である。

【図１２】

受け取られたドキュメントのマーケットメーカーノードプロセシングのフローチャートである。

【図１３】

本発明によるマーケットメーカーノードにおける参加者の登録のプロセスを説明するフローチャートである。

【図１４】

図９のプロセスによるマーケットメーカーノードにおけるサービスの仕様書の提供のプロセスを説明するフローチャートである。

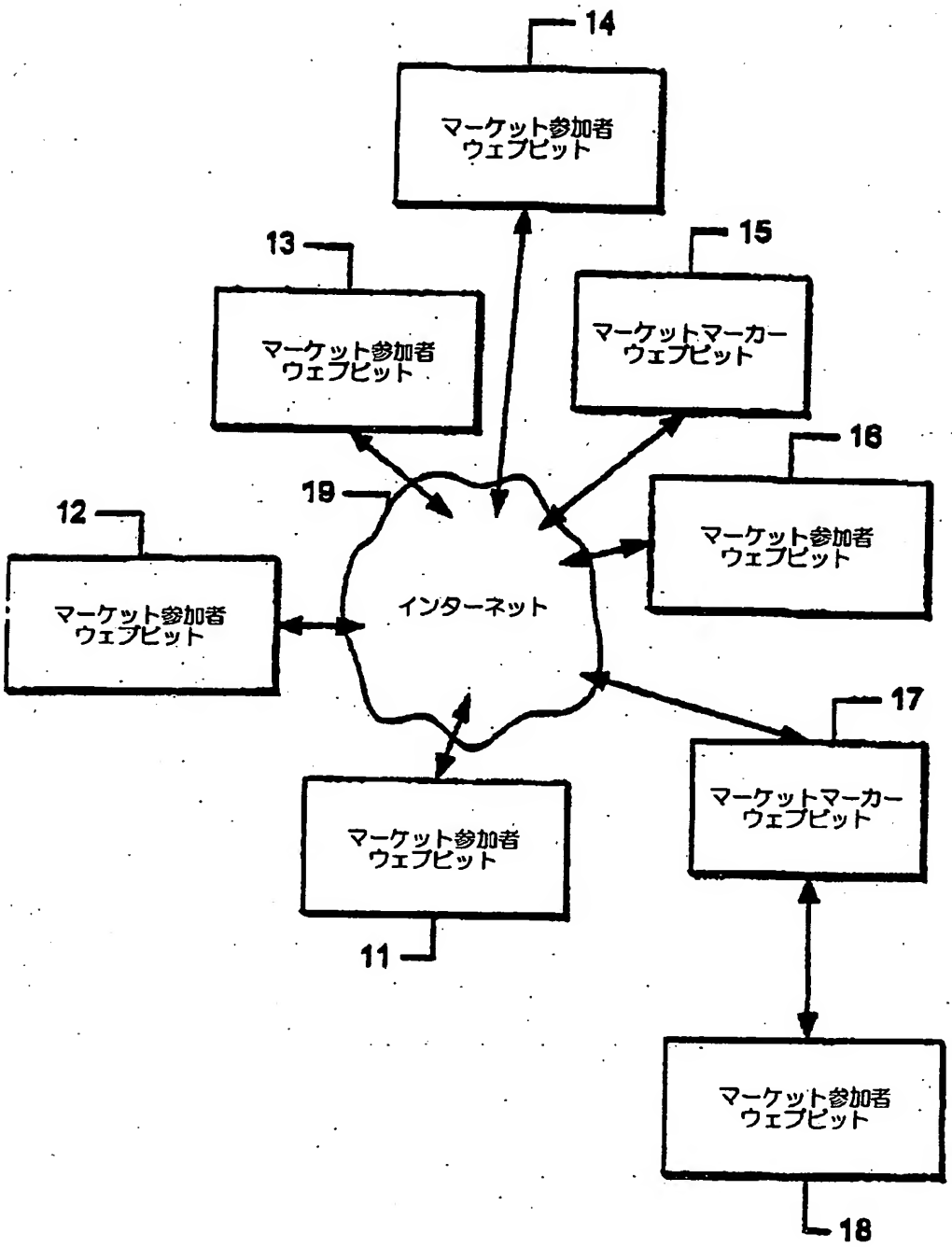
【図１５】

本発明によるマーケットメーカーまたは参加者における操作のシーケンスを説明するダイアグラムである。

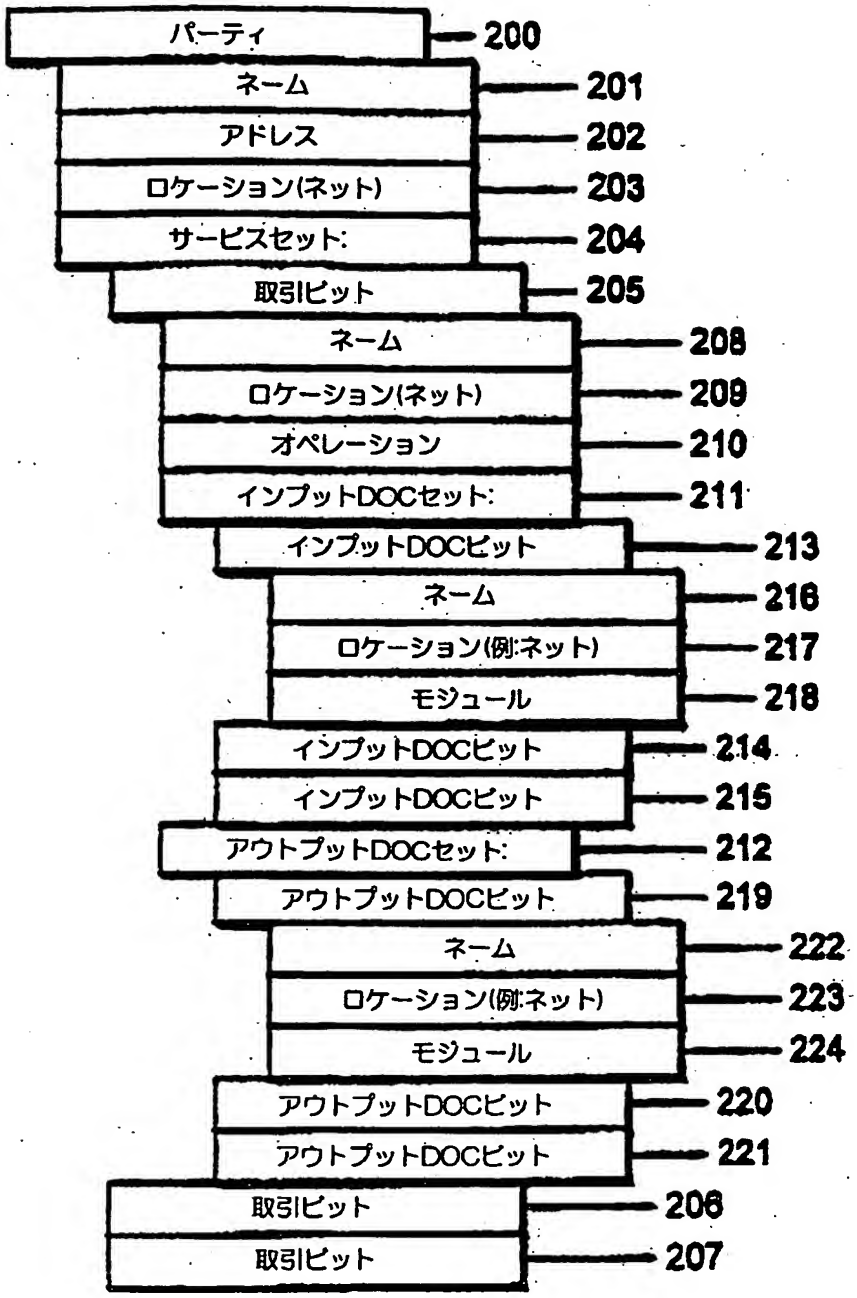
【図１６】

本発明によるＢＩＤに基づく商取引ネットワークの要素の概念的なダイアグラムである。

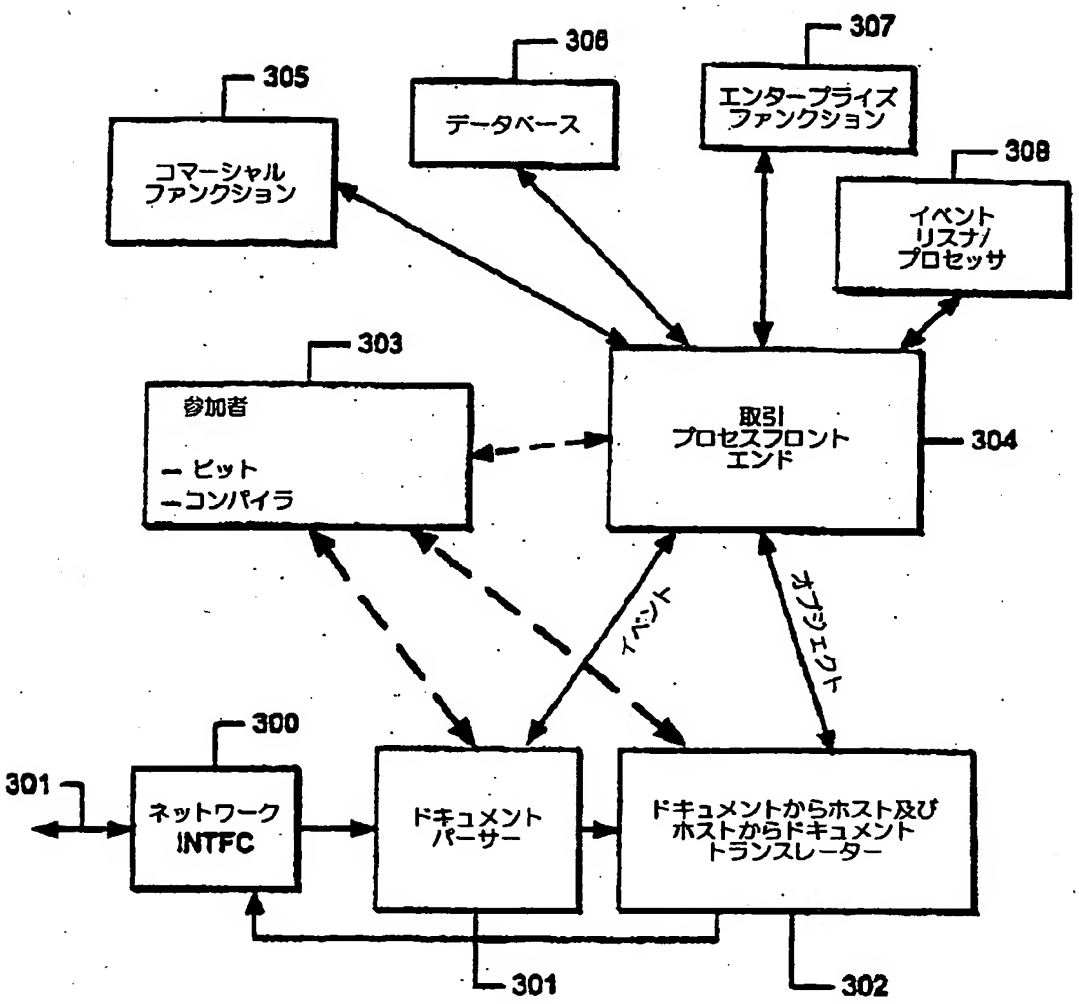
【図１】


14
マーケット参加者
ウェブビット
13
マーケット参加者
ウェブビット
15
マーケットマーカー
ウェブビット
16
マーケット参加者
ウェブビット
19
12
マーケット参加者
ウェブビット
インターネット
17
マーケットマーカー
ウェブビット
マーケット参加者
ウェブビット
11
マーケット参加者
ウェブビット
18

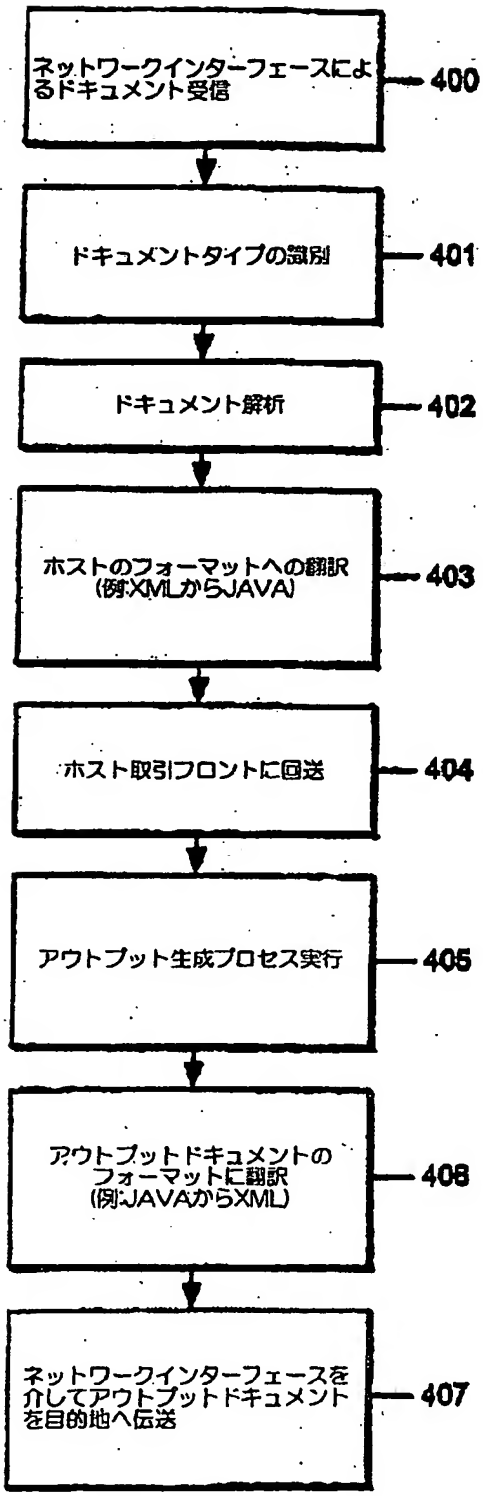
【図２】


パーティ
200
ネーム
201
アドレス
202
ロケーション(ネット)
203
サービスセット:
204
取引ビット
205
ネーム
208
ロケーション(ネット)
209
オペレーション
210
インプットDOCセット:
211
インプットDOCビット
213
ネーム
216
ロケーション(例:ネット)
217
モジュール
218
インプットDOCビット
214
インプットDOCビット
215
アウトプットDOCセット:
212
アウトプットDOCビット
219
ネーム
222
ロケーション(例:ネット)
223
モジュール
224
アウトプットDOCビット
220
アウトプットDOCビット
221
取引ビット
206
取引ビット
207

【図３】


305
コマーシャル
ファンクション
306
データベース
307
エンタープライズ
ファンクション
308
イベント
リスナ/
プロセッサ
303
参加者
－ビット
－コンパイラ
取引
プロセスフロント
エンド
304
イベント
オブジェクト
300
301
ネットワーク
INTFC
ドキュメント
パーサー
ドキュメントからホスト及び
ホストからドキュメント
トランスレーター
301
302

【図４】


ネットワークインターフェースによるドキュメント受信
400
ドキュメントタイプの識別
401
ドキュメント解析
402
ホストのフォーマットへの翻訳
(例:XMLからJAVA)
403
ホスト取引フロントに回送
404
アウトプット生成プロセス実行
405
アウトプットドキュメントのフォーマットに翻訳
(例:JAVAからXML)
406
ネットワークインターフェースを介してアウトプットドキュメントを目的地へ伝送
407

【図５】

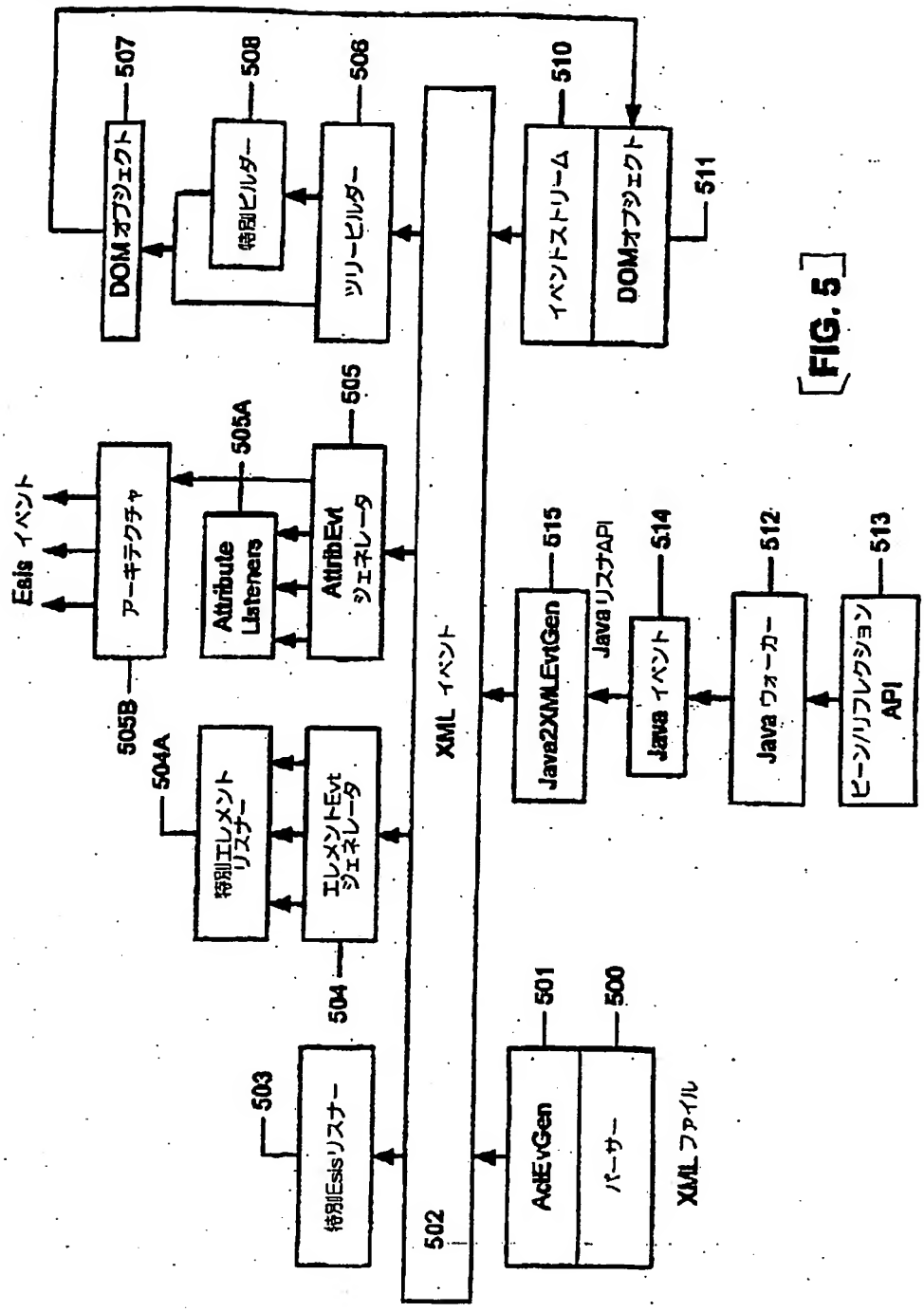

DOMオブジェクト
507
特別ビルダー
508
ツリービルダー
506
イベントストリーム
510
DOMオブジェクト
511
Esis イベント
アーキテクチャ
505B
505A
Attribute Listeners
AttribEvt ジェネレータ
505
504A
特別エレメントリスナー
エレメントEvt ジェネレータ
504
XML イベント
Java2XMLEvtGen
515
Java リスナAPI
Java イベント
514
Java ウォーカー
512
ビーンリフレクション API
513
503
特別Esisリスナー
501
AdiEvGen
500
パーサー
XML ファイル
502

FIG. 5

【図６】

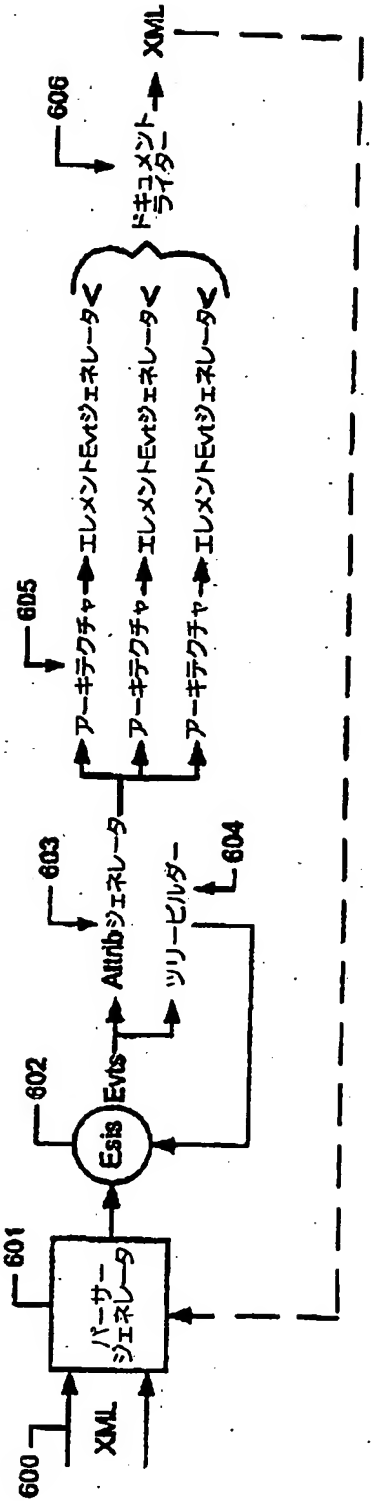
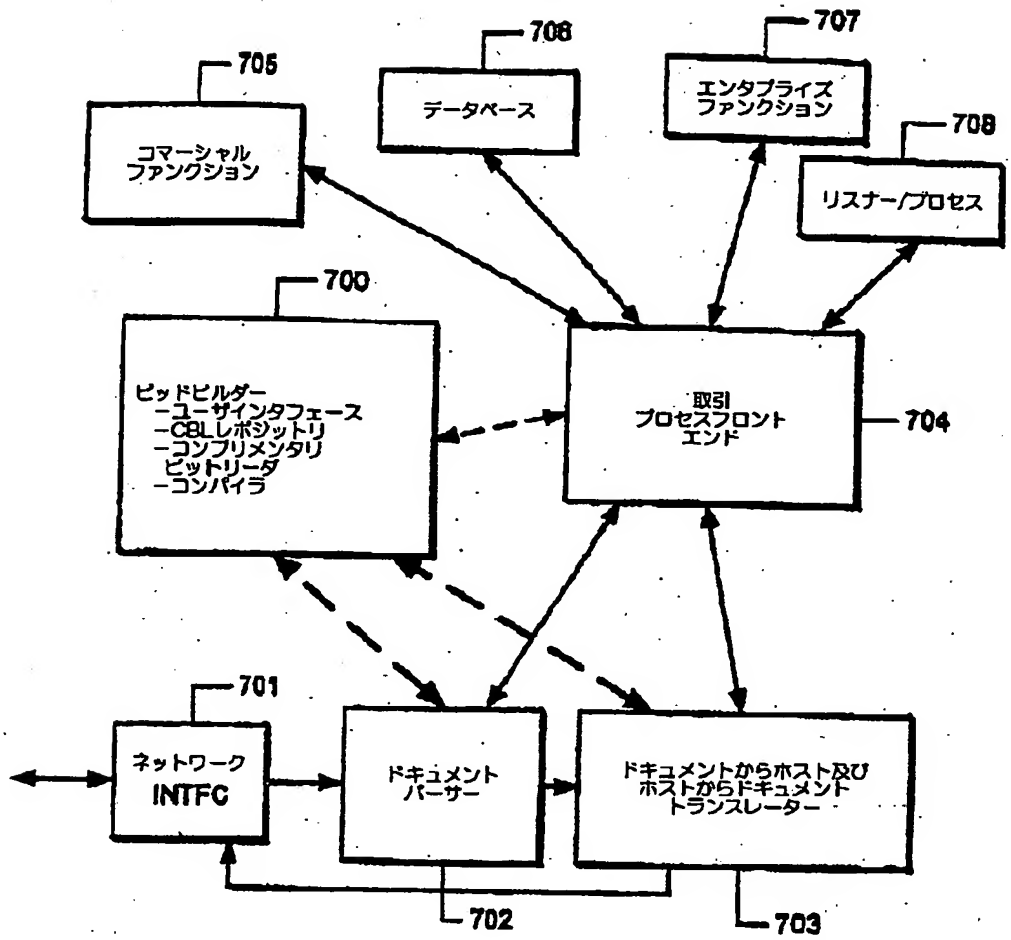
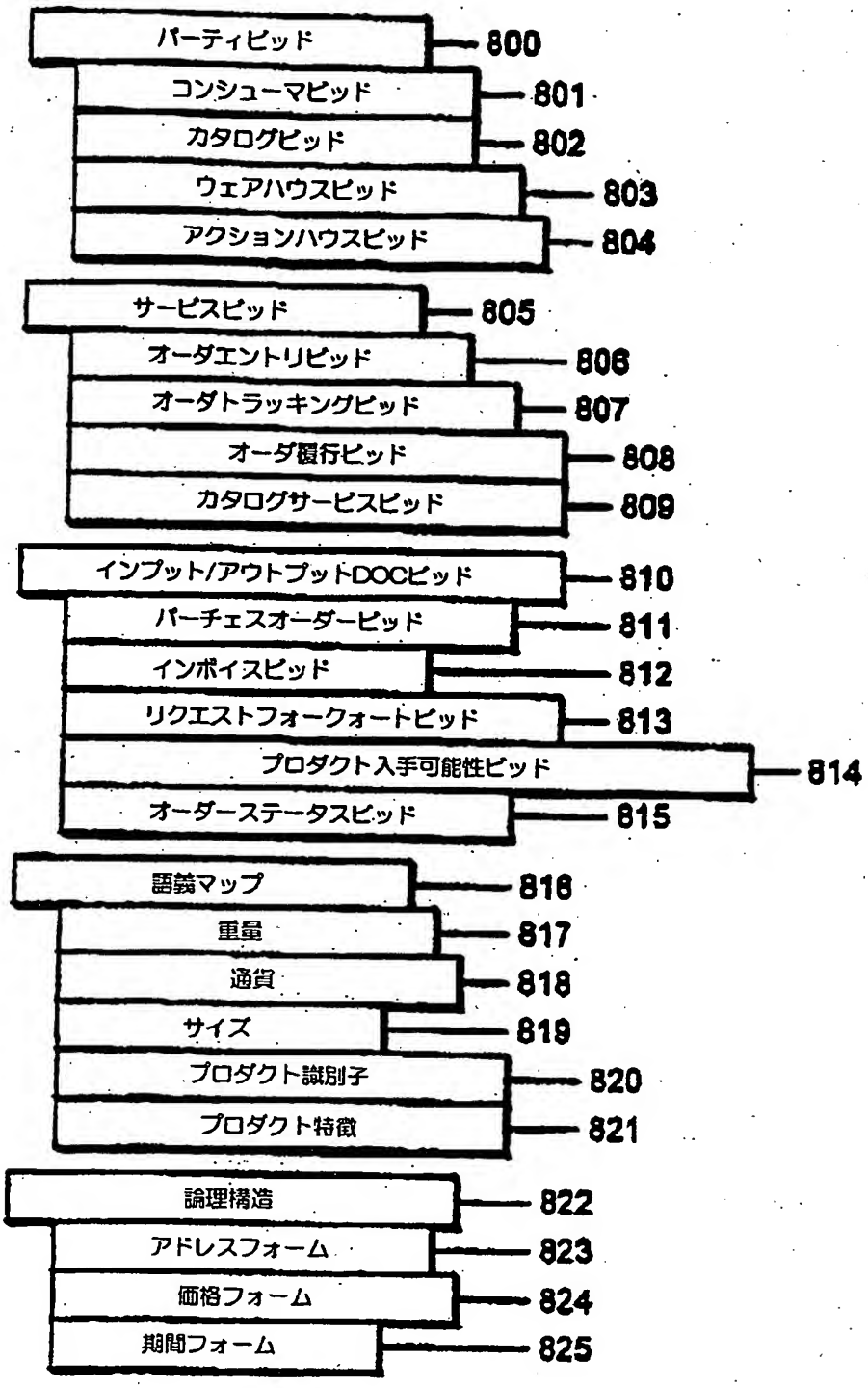

600
XML
601
パーサー
ジェネレータ
602
Esis
Evts
603
Attrbジェネレータ
ツリービルダー
604
605
アーキテクチャ
エレメントEvtジェネレータ
アーキテクチャ
エレメントEvtジェネレータ
アーキテクチャ
エレメントEvtジェネレータ
606
ドキュメント
ライター
XML

FIG. 6

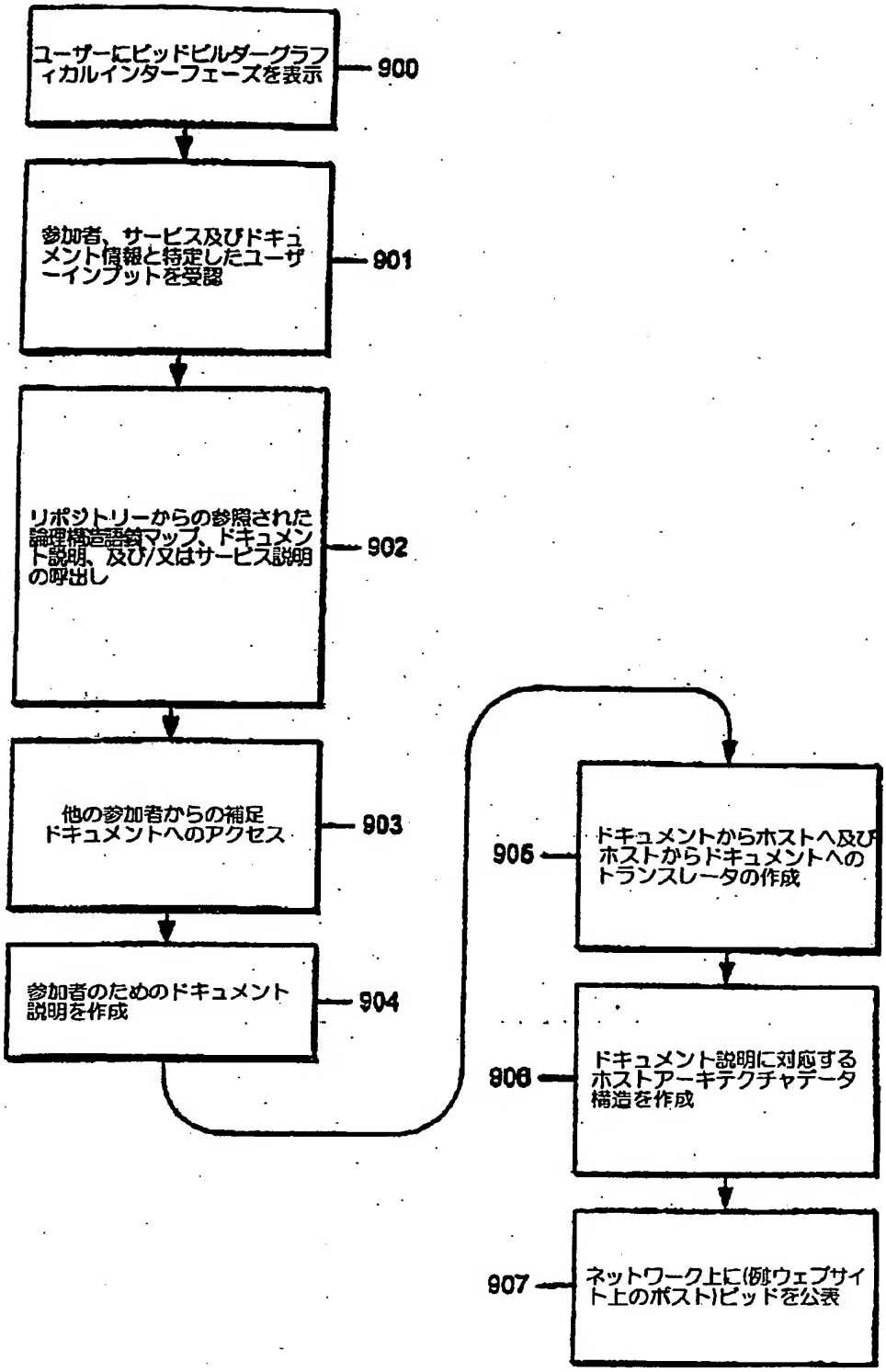
【図７】


705
コマーシャル
ファンクション
708
データベース
707
エンタプライズ
ファンクション
708
リスナー/プロセス
700
ビッドビルダー
－ユーザインタフェース
－C&Lレポジットリ
－コンプリメンタリ
ビットリーダ
－コンパイラ
取引
プロセスフロント
エンド
704
701
ネットワーク
INTFC
ドキュメント
パーサー
ドキュメントからホスト及び
ホストからドキュメント
トランスレーター
702
703

【図８】


パーティビッド
800
コンシューマビッド
801
カタログビッド
802
ウェアハウスビッド
803
アクションハウスビッド
804
サービスビッド
805
オーダエントリビッド
806
オーダトラッキングビッド
807
オーダ履行ビッド
808
カタログサービスビッド
809
インプット/アウトプットDOCビッド
810
パーチェスオーダービッド
811
インボイスビッド
812
リクエストフォークォートビッド
813
プロダクト入手可能性ビッド
814
オーダーステータスビッド
815
語義マップ
816
重量
817
通貨
818
サイズ
819
プロダクト識別子
820
プロダクト特徴
821
論理構造
822
アドレスフォーム
823
価格フォーム
824
期間フォーム
825

【図９】


ユーザーにビッドビルダーグラフィカルインターフェーズを表示
900
参加者、サービス及びドキュメント情報と特定したユーザーインプットを受認
901
リポジトリーからの参照された論理構造語義マップ、ドキュメント説明、及び/又はサービス説明の呼出し
902
他の参加者からの補足ドキュメントへのアクセス
903
参加者のためのドキュメント説明を作成
904
ドキュメントからホストへ及びホストからドキュメントへのトランスレータの作成
905
ドキュメント説明に対応するホストアーキテクチャデータ構造を作成
906
ネットワーク上に(例ウェブサイト上のポストビッドを公表
907

【図１０】

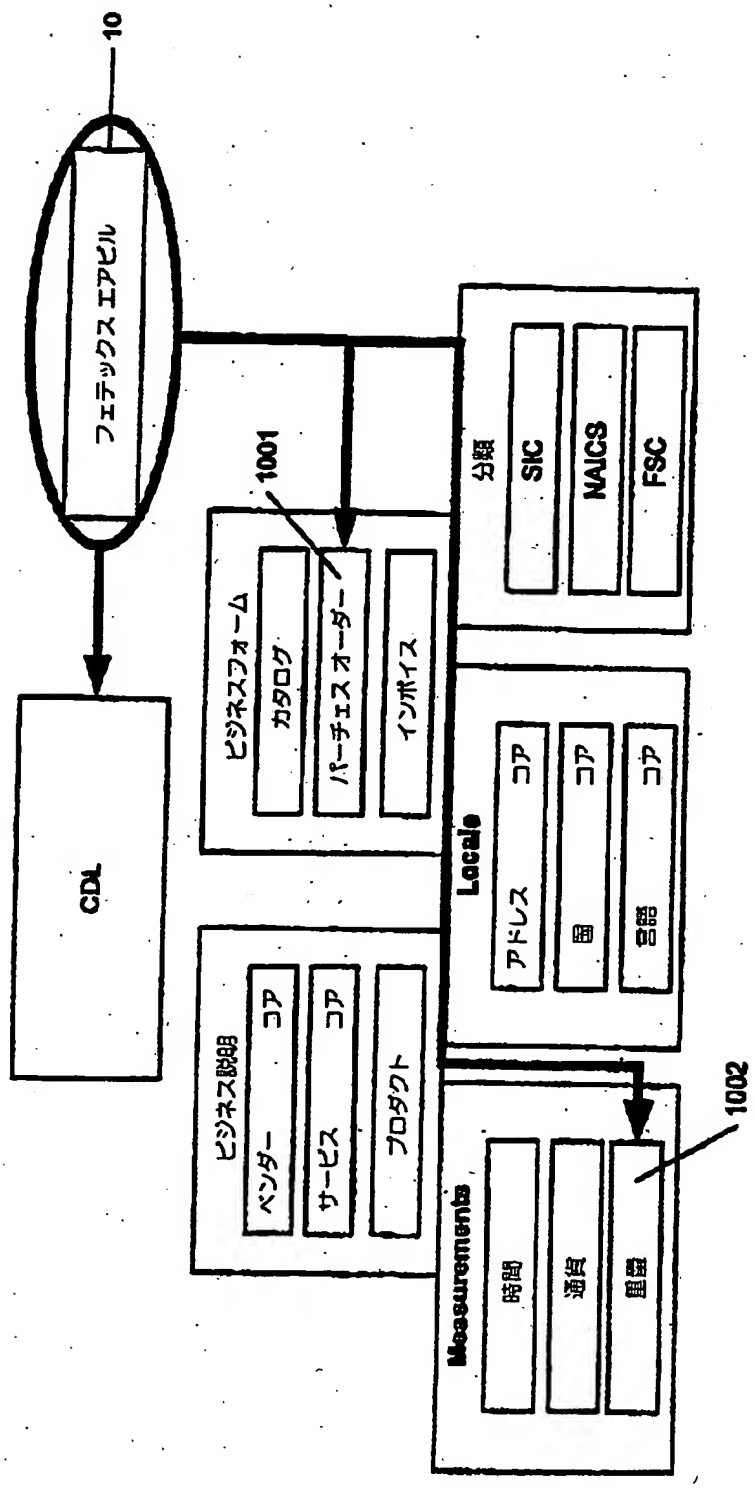
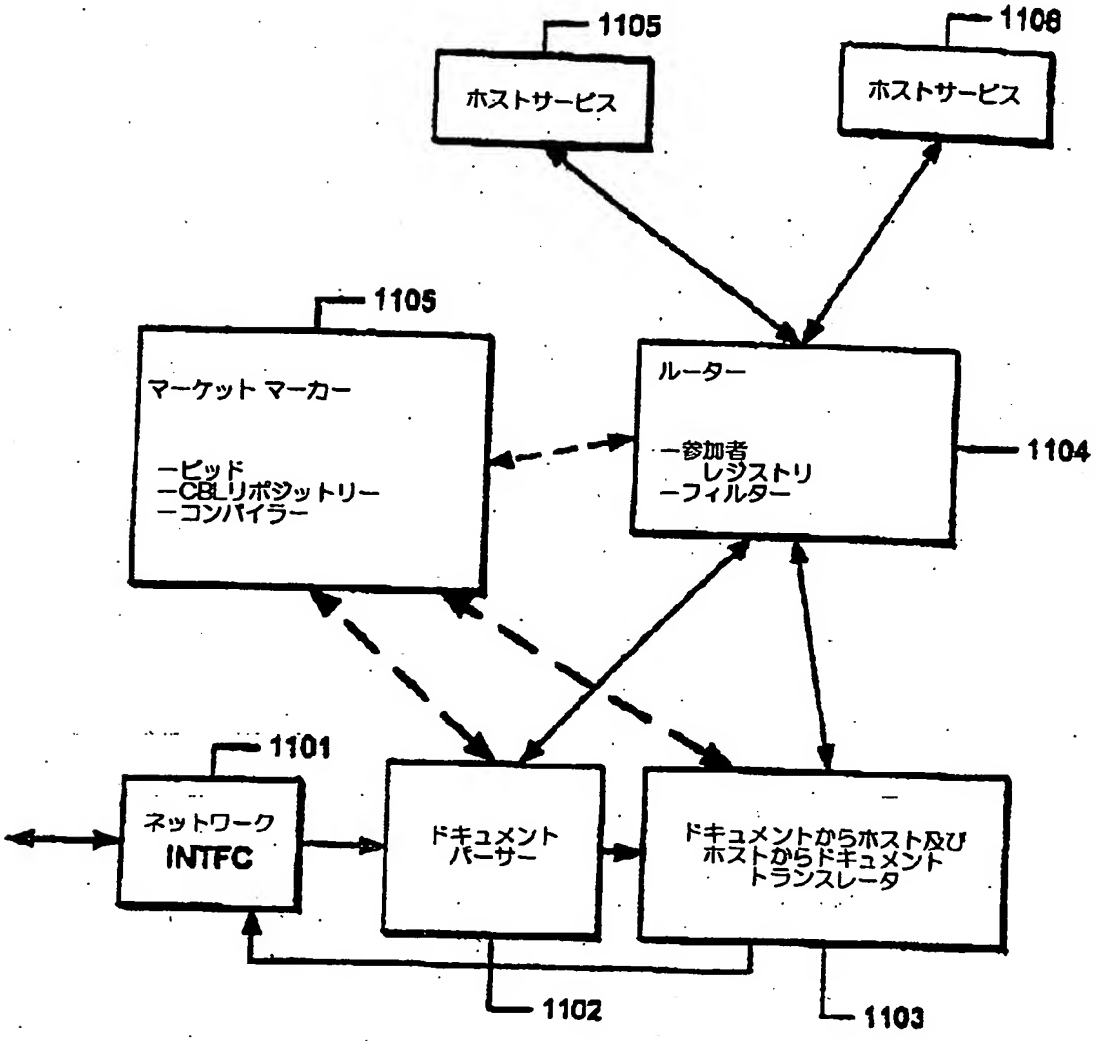
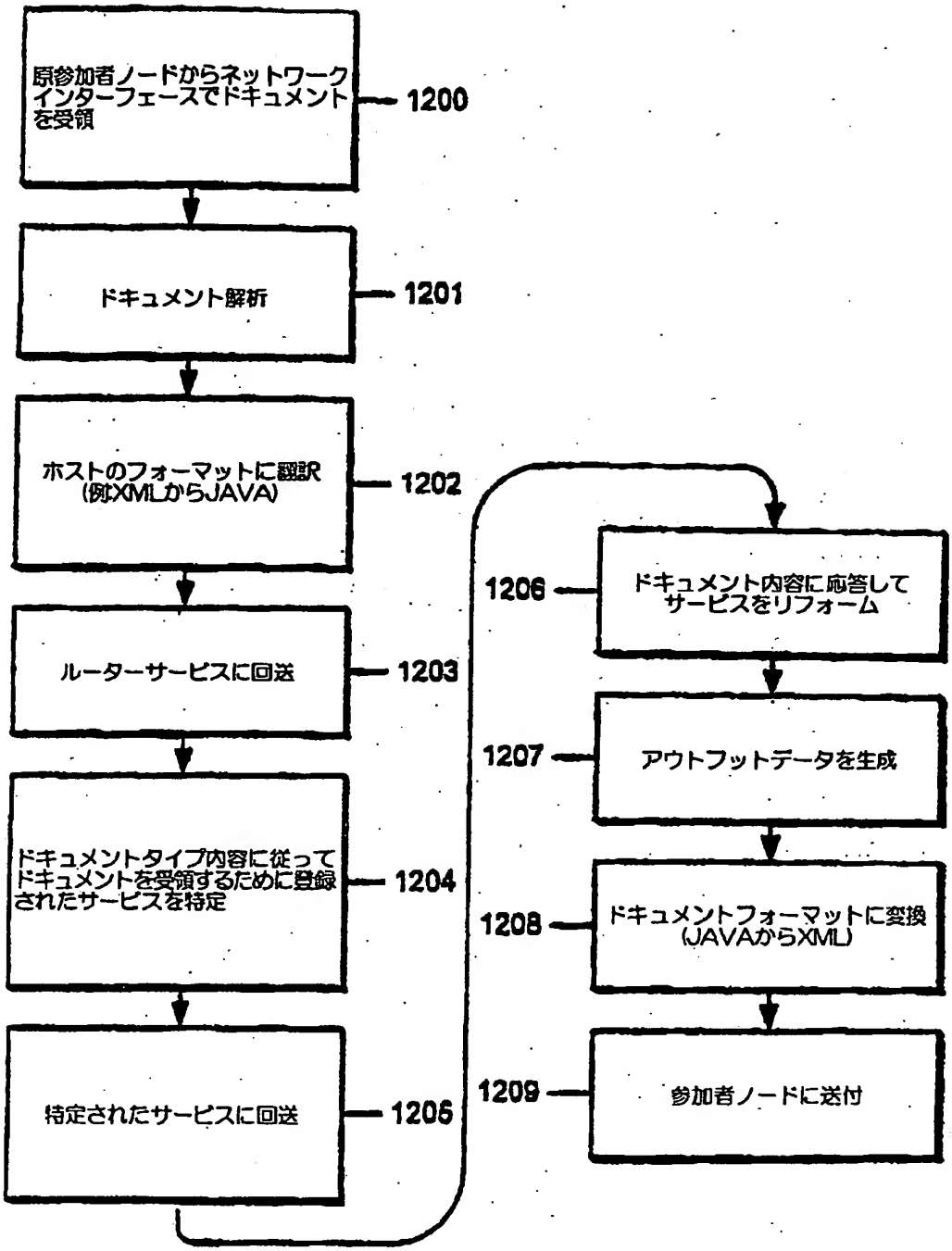

10
フェデックス エアビル
CDL
1001
ビジネスフォーム
カタログ
パーチェス オーダー
インボイス
分類
SIC
NAICS
FSC
Locale
アドレス
コア
国
コア
名前
コア
ビジネス説明
ベンダー
コア
サービス
コア
プロダクト
1002
Measurements
時間
通貨
重量

FIG. 10

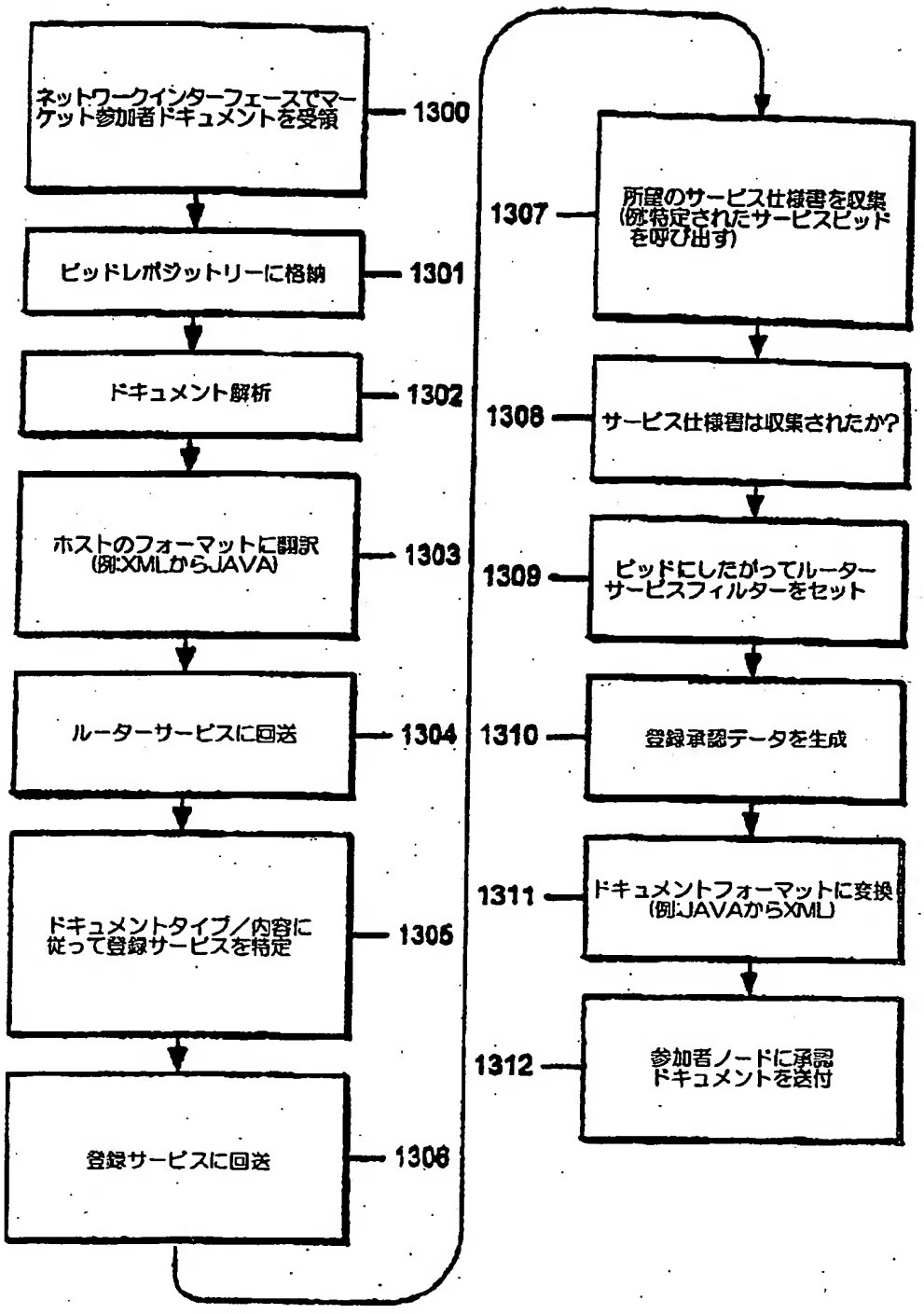
【図１１】


1105
ホストサービス
1106
ホストサービス
1105
マーケット マーカー
－ビッド
－CBLリポジットリー
－コンパイラー
ルーター
－参加者
レジストリ
－フィルター
1104
1101
ネットワーク
INTFC
ドキュメント
パーサー
ドキュメントからホスト及び
ホストからドキュメント
トランスレータ
1102
1103

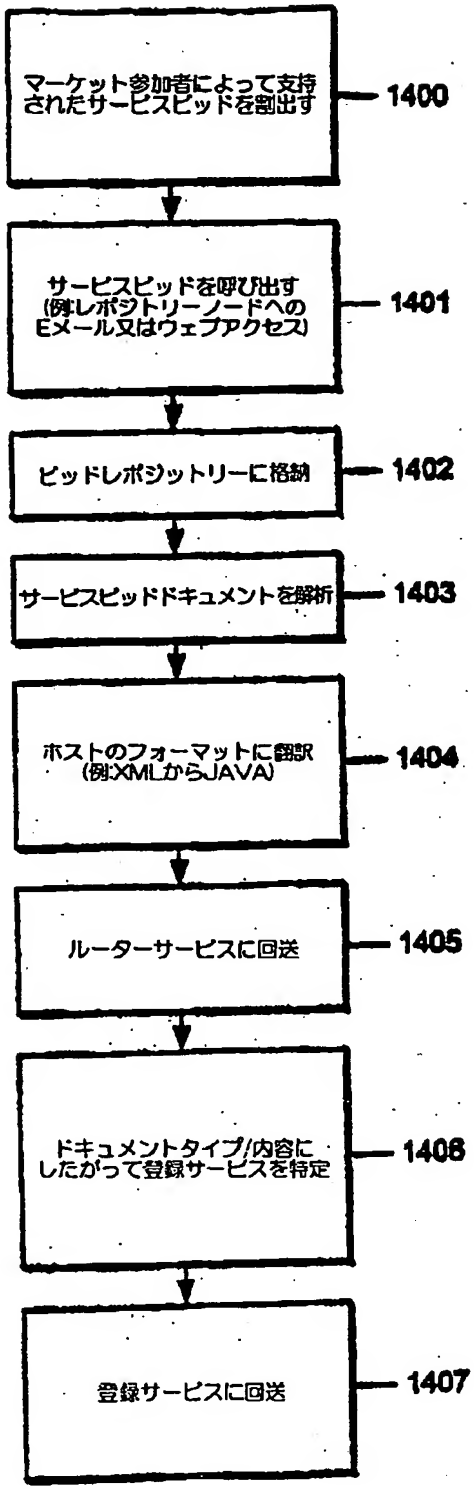
【図１２】


原参加者ノードからネットワークインターフェースでドキュメントを受領
1200
ドキュメント解析
1201
ホストのフォーマットに翻訳(例XMLからJAVA)
1202
ルーターサービスに回送
1203
ドキュメントタイプ内容に従ってドキュメントを受領するために登録されたサービスを特定
1204
特定されたサービスに回送
1205
ドキュメント内容に応答してサービスをリフォーム
1206
アウトフットデータを生成
1207
ドキュメントフォーマットに変換(JAVAからXML)
1208
参加者ノードに送付
1209

【図１３】


ネットワークインターフェースでマーケット参加者ドキュメントを受領
1300
ビッドレポジットリーに格納
1301
ドキュメント解析
1302
ホストのフォーマットに翻訳(例:XMLからJAVA)
1303
ルーターサービスに回送
1304
ドキュメントタイプ/内容に従って登録サービスを特定
1305
登録サービスに回送
1306
所望のサービス仕様書を収集(例:特定されたサービスビッドを呼び出す)
1307
サービス仕様書は収集されたか?
1308
ビッドにしたがってルーターサービスフィルターをセット
1309
登録承認データを生成
1310
ドキュメントフォーマットに変換(例:JAVAからXML)
1311
参加者ノードに承認ドキュメントを送付
1312

【図１４】


マーケット参加者によって支持されたサービスビッドを割出す
1400
サービスビッドを呼び出す(例レポジトリーノードへのEメール又はウェブアクセス)
1401
ビッドレポジットリーに格納
1402
サービスビッドドキュメントを解析
1403
ホストのフォーマットに翻訳(例XMLからJAVA)
1404
ルーターサービスに回送
1405
ドキュメントタイプ/内容にしたがって登録サービスを特定
1406
登録サービスに回送
1407

【図１５】

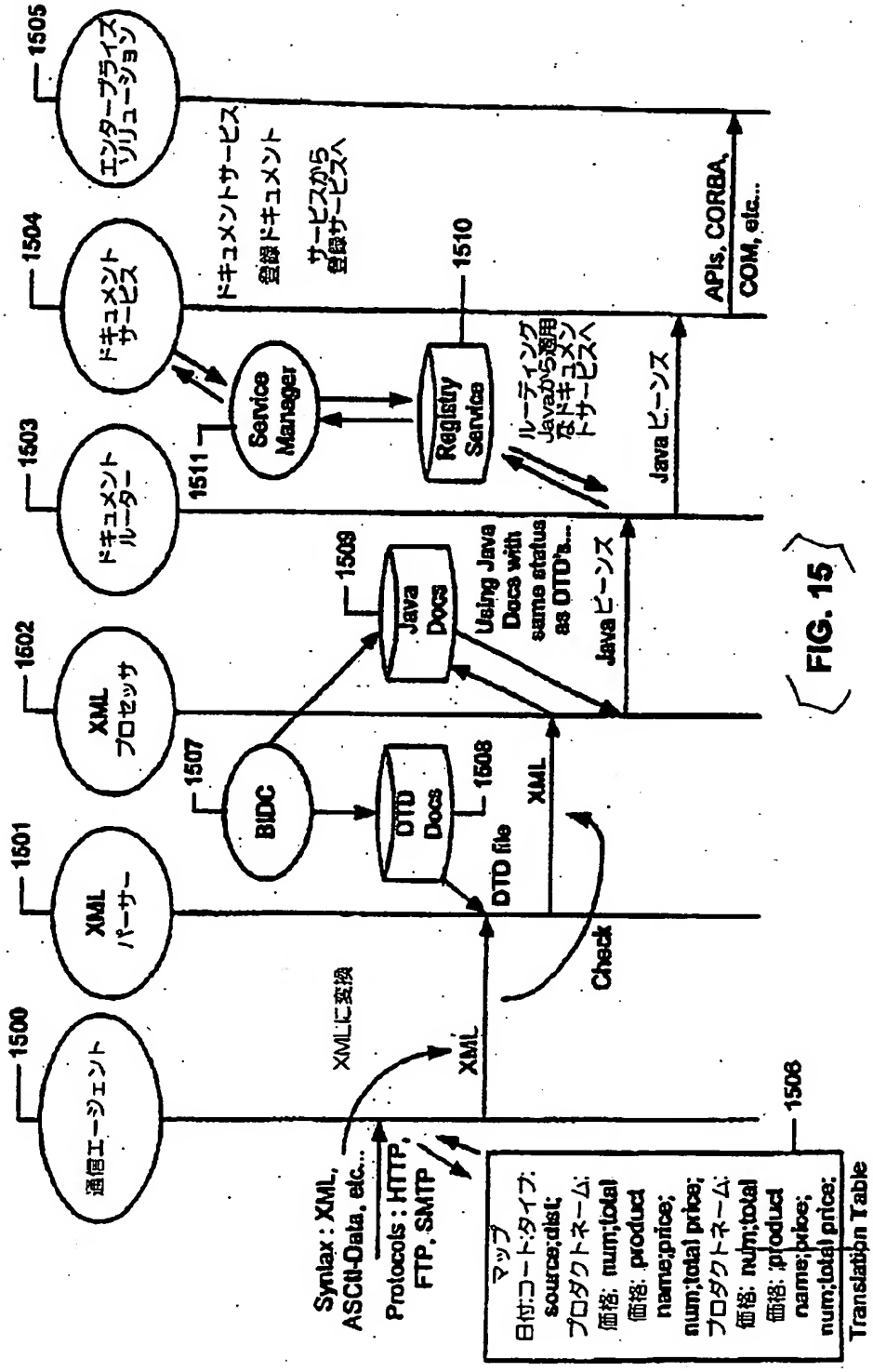
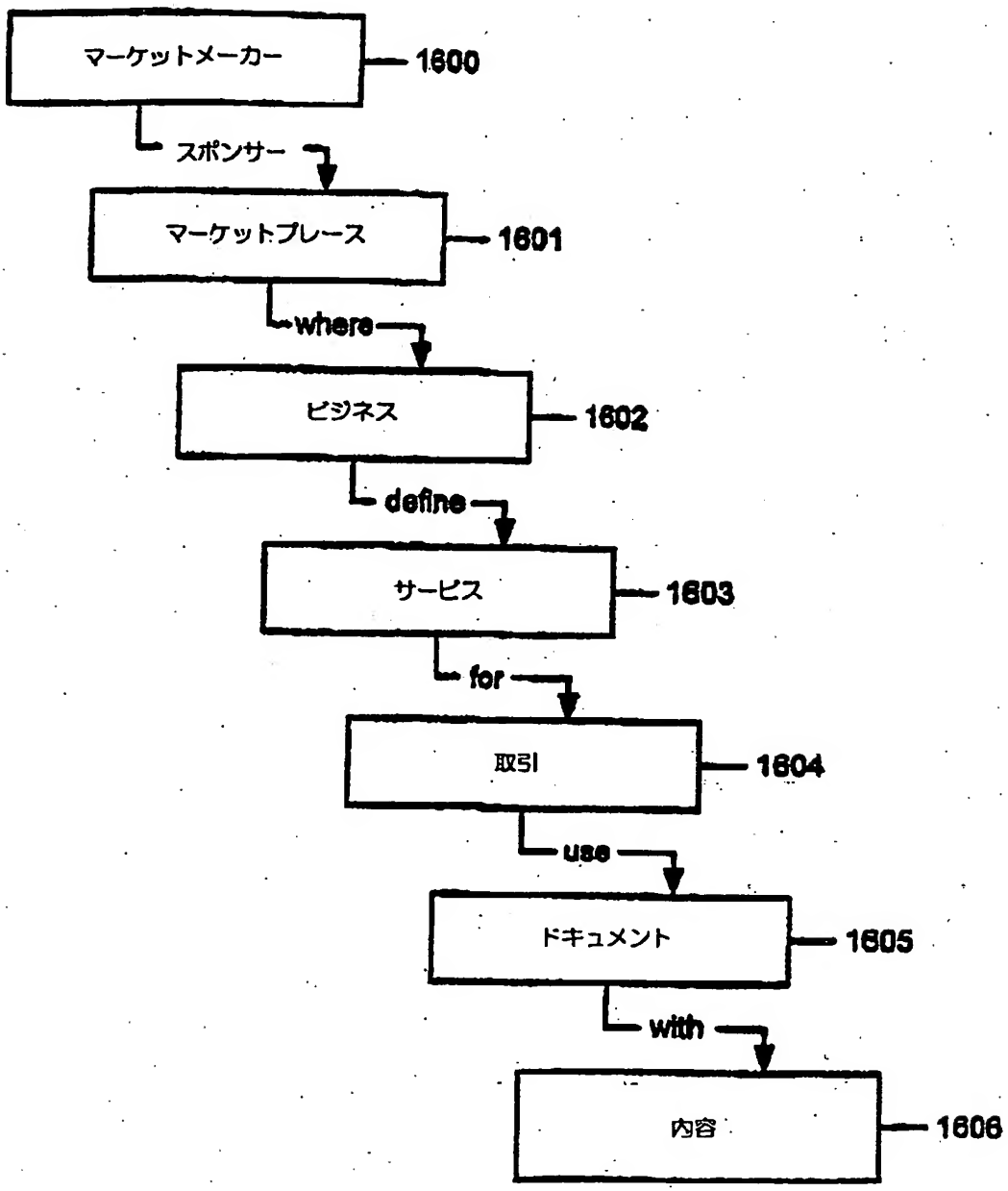

1500
通信エージェント
1501
XML パーサー
1502
XML プロセッサ
1503
ドキュメント ルーター
1504
ドキュメント サービス
1505
エンタープライズ ソリューション
XML に変換
Syntax : XML, ASCII-Data, etc...
Protocols : HTTP, FTP, SMTP
XML
Check
1506
マップ
日付:コード:タイプ:
source;dist;
プロダクトネーム:
価格: num;total
価格: product
name;price;
num;total price;
プロダクトネーム:
価格: num;total
価格: product
name;price;
num;total price;
Translation Table
1507
BIDC
1508
DTD Docs
DTD file
XML
1509
Java Docs
Using Java Docs with same status as DTD's...
Java ビーンズ
1510
Registry Service
ルーティング Javaから適用 なドキュメン トサービスへ
1511
Service Manager
ドキュメントサービス
登録ドキュメント
サービスから
登録サービスへ
Java ビーンズ
APIs, CORBA, COM, etc...

FIG. 15

【図１６】


マーケットメーカー
1600
スポンサー
マーケットプレース
1601
where
ビジネス
1602
define
サービス
1603
for
取引
1604
use
ドキュメント
1605
with
内容
1606

【国際調査報告】

INTERNATIONAL SEARCH REPORT

Inter nal Application No
PCT/US 99/23426

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/60

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	DUDECK J: "Aspects of implementing and harmonizing healthcare communication standards" INTERNATIONAL JOURNAL OF MEDICAL INFORMATICS,IR,ELSEVIER SCIENTIFIC PUBLISHERS, SHANNON, vol. 48, no. 1-3, 1 February 1998 (1998-02-01), pages 163-171, XP004115318 ISSN: 1386-5056 abstract page 169, column 1, line 33 -page 170, column 2, line 18 ---- -/--	1-258

[X] Further documents are listed in the continuation of box C. [X] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
18 April 2000	27/04/2000
Name and mailing address of the ISA European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Authorized officer Suendermann, R

Form PCT/ISA/210 (second sheet) (July 1992)

page 1 of 2

INTERNATIONAL SEARCH REPORT

Inter national Application No
PCT/US 99/23426

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	KRISTENSEN A: "Template resolution in XML/HTML" COMPUTER NETWORKS AND ISDN SYSTEMS,NL,NORTH HOLLAND PUBLISHING. AMSTERDAM, vol. 30, no. 1-7, 1 April 1998 (1998-04-01), pages 239-249, XP004121423 ISSN: 0169-7552 abstract page 239, column 1, line 1 -page 248, column 2, line 23	1-258
A	WO 98 34179 A (TIME BASE PTY LIMITED ;MARIANI PETER (AU); LESSING ABHA (AU); SCHN) 6 August 1998 (1998-08-06) abstract; claims 1,6,25,45 page 4, line 25 -page 5, line 12 page 6, line 1 -page 10, line 20	1,17,39, 61,73, 98,212, 236
A	LIECHTI O ET AL: "Structured graph format: XML metadata for describing Web site structure" COMPUTER NETWORKS AND ISDN SYSTEMS,NL,NORTH HOLLAND PUBLISHING. AMSTERDAM, vol. 30, no. 1-7, 1 April 1998 (1998-04-01), pages 11-21, XP004121444 ISSN: 0169-7552 abstract page 11, column 1, line 1 -page 209, column 1, line 5	1,17,39, 61,73, 98,212, 236

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

information on patent family members

Inter... ...nal Application No
PCT/US 99/23426

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9834179 A	06-08-1998	AU 706144 B	10-06-1999
		AU 5741498 A	25-08-1998
		EP 0954808 A	10-11-1999

Form PCT/ISA/210 (patent family annex) (July 1992)

フロントページの続き

(51)Int.Cl.[7]	識別記号	FI	テーマコード(参考)
G06F 17/60	ZEC	G06F 17/60	ZEC

(31)優先権主張番号 09／173，854
(32)優先日 平成10年10月16日(1998．10．16)
(33)優先権主張国 米国(US)
(81)指定国 EP(AT，BE，CH，CY，DE，DK，ES，FI，FR，GB，GR，IE，IT，LU，MC，NL，PT，SE)，OA(BF，BJ，CF，CG，CI，CM，GA，GN，GW，ML，MR，NE，SN，TD，TG)，AP(GH，GM，KE，LS，MW，SD，SL，SZ，TZ，UG，ZW)，EA(AM，AZ，BY，KG，KZ，MD，RU，TJ，TM)，AE，AL，AM，AT，AU，AZ，BA，BB，BG，BR，BY，CA，CH，CN，CR，CU，CZ，DE，DK，DM，EE，ES，FI，GB，GD，GE，GH，GM，HR，HU，ID，IL，IN，IS，JP，KE，KG，KP，KR，KZ，LC，LK，LR，LS，LT，LU，LV，MD，MG，MK，MN，MW，MX，NO，NZ，PL，PT，RO，RU，SD，SE，SG，SI，SK，SL，TJ，TM，TR，TT，UA，UG，US，UZ，VN，YU，ZA，ZW

(72)発明者 アーレン テリー
アメリカ合衆国 カリフォルニア州 95473 セバストポール セバストポール ロード 234
(72)発明者 フークス マシュー ダニエル
アメリカ合衆国 カリフォルニア州 95032 ロス ガトス ジャカランダ ウェイ 16229
(72)発明者 グルスコ ロバート ジョン
アメリカ合衆国 カリフォルニア州 94116 サン フランシスコ メンドーザ アベニュー 159
(72)発明者 マロニー マーレイ
カナダ オンタリオ エル1ダブリュー 3ケイ6 ピッカリング カウアン サークル 671
(72)発明者 ディヴィッドソン アンドリュー エヴァレット
アメリカ合衆国 カリフォルニア州 95006 ボールダー クリーク ホプキンス ガルチ 1222
(72)発明者 パーソン ケネス
アメリカ合衆国 カリフォルニア州 95060 サンタ クルーズ ヴィレッジ サークル 500

(72)発明者 シュヴァーツホッフ ケリー レーン
アメリカ合衆国 カリフォルニア州
94306 パロ アルト アラストラデロ
ロード 535 アパートメント 204

Fターム(参考) 5B082 EA01 GA06 HA07

【公報種別】特許法第１７条第１項及び特許法第１７条の２の規定による補正の掲載
【部門区分】第６部門第３区分
【発行日】平成１５年２月２５日（２００３．２．２５）

【公表番号】特表２００２－５２８７９７（Ｐ２００２－５２８７９７Ａ）
【公表日】平成１４年９月３日（２００２．９．３）
【年通号数】
【出願番号】特願２０００－５７７５９８（Ｐ２０００－５７７５９８）
【国際特許分類第７版】
G06F 19/00 300
12/00 546
547
17/60 306
310
ZEC
【ＦＩ】
G06F 19/00 300 N
12/00 546 A
547 H
17/60 306
310 Z
ZEC

【手続補正書】
【提出日】平成１４年４月５日（２００２．４．５）
【手続補正１】
【補正対象書類名】明細書
【補正対象項目名】特許請求の範囲
【補正方法】変更
【補正の内容】
【特許請求の範囲】
【請求項１】　トランザクション（transaction）に関連する処理を実行する複数のノード（nodes）を含むネットワークにおけるノード間のトランザクションのためのインターフェース（interface）であって、
入力ドキュメント（document）の定義及び出力ドキュメントの定義を供給する翻訳情報を含む、ネットーワークにおける少なくとも一つのノードによってアクセス可能なメモリに記録されたトランザクション処理へのインターフェースの機械可読仕様（machine readable specification）であり、前記入力及び出力ドキュメントの定義は、記録装置の組の個別の記述と、及び前記記録装置の組に対する論理構造を具備する前記仕様を具備することを特徴とするインターフェース。
【請求項２】　前記翻訳情報は、前記入力及び出力ドキュメントの定義における少なくとも一つの論理構造に対するデータ・タイプ（data type）仕様を含むことを特徴とする請求項１に記載のインターフェース。
【請求項３】　前記翻訳情報は、前記入力及び出力ドキュメントの定義における特定の論理構造に対する予め決められた組の記録装置を、リスト（list）の個別のエントリ（entries）にマッピング（mapping）する少なくとも一つのデータ構造を含むことを特徴とする請求項１に記載のインターフェース。
【請求項４】　論理構造のライブラリと、及び論理構造に対する翻訳情報を記録する、ネットワークにおける少なくとも一つのノードによってアクセス可能であるメモリにおけるリポジトリを含むことを特徴とする請求項１に記載のインターフェース。
【請求項５】　前記機械可読仕様は、特定のトランザクションの識別子を記

録するための論理構造と、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項１に記載のインターフェース。

【請求項６】　前記機械可読仕様は、インターフェースの識別子を記録するための、及び前記インターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項１に記載のインターフェース。

【請求項７】　前記機械可読仕様は、特定のトランザクションの仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項６に記載のインターフェース。

【請求項８】　前記記録装置はパースされた（parsed）データを具備することを特徴とする請求項１に記載のインターフェース。

【請求項９】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタ（text character）をコード化するキャラクタ・データ（character data）と；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識別するマーク付けデータとを具備することを特徴とする請求項８に記載のインターフェース。

【請求項１０】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項９に記載のインターフェース。

【請求項１１】　前記入力及び出力ドキュメントの少なくとも一つの特定の論理構造によって識別される前記記録装置の組の少なくとも一つに対する前記翻訳情報は、パースされたキャラクタの組に対する個別の定義をコード化すること

を特徴とする請求項８に記載のインターフェース。

【請求項１２】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項８に記載のインターフェース。

【請求項１３】　複数のトランザクションにおける使用のためのドキュメント・タイプのネットワークにおいて、少なくとも一つのノードによってアクセス可能であるメモリに記録されたレポジトリを含み、及び前記入力及び出力ドキュメントの一つの定義は、前記レポジトリにおけるドキュメント・タイプへの参照を含むことを特徴とする請求項１に記載のインターフェース。

【請求項１４】　ドキュメント・タイプの前記レポジトリは、ネットワークにおける参加者処理（participant processes）を識別するためのドキュメント・タイプを含むことを特徴とする請求項１３に記載の方法。

【請求項１５】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項１に記載のインターフェース。

【請求項１６】　翻訳情報を含む前記機械可読データ構造は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義に従って編成されたドキュメントを具備することを特徴とする請求項１に記載のインターフェース。

【請求項１７】　ネットワーク・インターフェースと、及びトランザクション処理アーキテクチャに従って、トランザクションのトランザクション処理を実行するデータ処理リソースを含むシステム上で実行されるトランザクションに対する参加者インターフェース（participant interface）を設定するための装置であって：

前記システムによって実行され、前記システムによってアクセス可能な媒体に記録され、特定のトランザクションにおける参加者に対する参加者インターフェースの定義を構築するためのツールを供給する命令のプログラムであって、前記参加者インターフェースの定義は、前記参加者に対する入力ドキュメントの定義と、及び前記参加者に対する出力ドキュメントの定義とを含み、前記入力及び出力ドキュメントの定義は、記録装置の組の個別の機械可読記述と、及び前記記録装置の組に対する論理構造とを具備する前記プログラムと；及び

前記システムによって実行可能であり、前記システムによってアクセス可能な媒体に記録され、前記記録装置の組に対応するデータ構造と、及び前記トランザクション処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造とをコンパイルするために、前記入力ドキュメントを、前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令をコンパイルするために、及び前記トランザクション処理の出力を、前記記録装置の組と及び前記出力ドキュメントの論理構造に翻訳するために、前記システムによって実行可能な命令をコンパイルするために、前記入力及び出力ドキュメントの定義に応答するプログラムとを具備することを特徴とする装置。

【請求項１８】　参加者インターフェースの定義を構築するための前記ツールは、レポジトリからの前記定義のエレメント（element）にアクセスするために前記システムによって実行可能である命令を含み、前記レポジトリは、論理構造のライブラリと、及びインターフェース記述を構築するために使用される論理構造に対する翻訳情報とを記録することを特徴とする請求項１７に記載の装置。

【請求項１９】　前記レポジトリは、論理構造を具備するドキュメントの定義を記録することを特徴とする請求項１８に記載の装置。

【請求項２０】　参加者インターフェースの定義を構築するための前記ツールは、

相補トランザクションに対する他の参加者インターフェースの定義にアクセスするためであり、前記アクセスされた定義は、前記相補トランザクションに対する入力ドキュメントの定義と、及び前記相補トランザクションに対する出力ドキュメントの定義を含む前記アクセスのために；及び

構築されている前記インターフェースの入力ドキュメントの定義のように、前記相補トランザクションの前記出力ドキュメントの定義を含むことによって、前記参加者インターフェースの定義を設定するために、前記システムによって実行可能な命令を含むことを特徴とする請求項１７に記載の装置。

【請求項２１】　参加者インターフェースの定義を構築するための前記ツールは、

構築されているインターフェースの前記出力ドキュメントの定義のように、前

記相補トランザクションの入力ドキュメントの定義を含むために、

前記システムによって実行可能である命令を含むことを特徴とする請求項２０に記載の装置。

【請求項２２】　参加者インターフェースの定義を構築するための前記ツールは、特定のトランザクションの識別子を記録するための論理構造を含む参加者インターフェース・ドキュメントの定義に従ったドキュメントと、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つとを構築するために、前記システムによって実行可能である命令を含むことを特徴とする請求項１７に記載の装置。

【請求項２３】　参加者インターフェースの定義を構築するための前記ツールは、前記参加者インターフェースの識別子を記録するため、及び前記参加者インターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含む参加者インターフェース・ドキュメントの定義に従ったドキュメントを構築するために、前記システムによって実行可能な命令を含むことを特徴とする請求項１７に記載の装置。

【請求項２４】　参加者インターフェース・ドキュメントの定義に従った前記ドキュメントは、特定のトランザクションの機械可読仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項２３に記載の装置。

【請求項２５】　前記記録装置は、パースされたデータを具備することを特徴とする請求項１７に記載の装置。

【請求項２６】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタ（text character）をコード化するキャラクタ・データ（character data）と；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識

別するマーク付けデータとを具備することを特徴とする請求項２５に記載の装置。

【請求項２７】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項２６に記載の装置。

【請求項２８】　前記仕様は、前記入力及び出力ドキュメントのうち少なくとも一つの前記論理構造によって識別される前記記録装置の組のうち少なくとも一つに対する翻訳情報を含み、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項２５に記載の装置。

【請求項２９】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項２５に記載の装置。

【請求項３０】　前記記録装置の組に対応するデータ構造及び前記入力及び出力ドキュメントの論理構造は、可変トランザクション処理アーキテクチャに従った変数及び方法を含むプログラミング・オブジェクトを含むことを特徴とする請求項１７に記載の装置。

【請求項３１】　前記トランザクション処理の前記可変トランザクション処理アーキテクチャは、インターフェース記述言語に従った処理を具備することを特徴とする請求項１７に記載の装置。

【請求項３２】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項１７に記載の装置。

【請求項３３】　前記入力及び出力ドキュメントの一つの前記定義は、レポジトリにおけるドキュメント・タイプへの参照を含むことを特徴とする請求項１９に記載の装置。

【請求項３４】　前記レポジトリは、トランザクションの製品サブジェクトの測定を特定する翻訳情報を含むことを特徴とする請求項１８に記載の装置。

【請求項３５】　前記レポジトリは、トランザクションの製品サブジェクトのコスト（cost）を特定する翻訳情報を含むことを特徴とする請求項１８に記載の装置。

【請求項３６】　前記レポジトリは、トランザクションの製品サブジェクトの特性を特定する翻訳情報を含むことを特徴とする請求項１８に記載の装置。

【請求項３７】　前記レポジトリは、トランザクションの金融ターム（financial terms）を特定する翻訳情報を含むことを特徴とする請求項１８に記載の装置。

【請求項３８】　前記レポジトリは、トランザクションの製品サブジェクトに対する出荷（shipment）のタームを特定する翻訳情報を含むことを特徴とする請求項１８に記載の装置。

【請求項３９】　システム上で実行されるトランザクションに対する参加者インターフェースを設定するための装置であって：

命令のデータ及びプログラムを記録するメモリと；

命令の前記プログラムを実行するメモリに接続されたデータ・プロセッサであって；前記命令のプログラムは、

特定のトランザクションにおける参加者のための参加者インターフェースの定義を構築するためのツールであって、参加者インターフェースの前記定義は、前記参加者に対する入力ドキュメントの定義及び前記参加者に対する出力ドキュメントの定義を含み、前記入力及び出力ドキュメントの定義は、記録装置の組の個別の機械可読記述と、及び前記記録装置の組に対する論理構造とを具備する前記ツールと；及び

前記入力及び出力ドキュメントの定義に応答し、前記記録装置の組と及び前記トランザクション処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造とに対応したデータ構造をコンパイルするため、前記入力ドキュメントを前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令をコンパイルし、及び前記トランザクション処理の出力を、前記記録装置の組と及び前記出力ドキュメントの論理構造とに翻訳するために、前記システムによって実行可能な命令をコンパイルするためのコンパイラとを含む前記データ・プロセッサとを具備することを特徴とする装置

【請求項４０】　前記データ・プロセッサによってアクセス可能なメモリに

記録されたレポジトリを含み、参加者インターフェースの定義を構築するための前記ツールは、レポジトリからの前記定義のエレメントにアクセスするために前記システムによって実行可能である命令を含み、前記レポジトリは、論理構造のライブラリと、及びインターフェース記述を構築するために使用される論理構造に対する翻訳情報とを記録することを特徴とする請求項３９に記載の装置。

【請求項４１】　前記レポジトリは、論理構造を具備するドキュメントの定義を記録することを特徴とする請求項４０に記載の装置。

【請求項４２】　参加者インターフェースの定義を構築するための前記ツールは、

相補トランザクションに対する他の参加者インターフェースの定義にアクセスするためであり、前記アクセスされた定義は、前記相補トランザクションに対する入力ドキュメントの定義と、及び前記相補トランザクションに対する出力ドキュメントの定義を含む前記アクセスのために；及び

構築されている前記インターフェースの入力ドキュメントの定義のように、前記相補トランザクションの前記出力ドキュメントの定義を含むことによって、前記参加者インターフェースの定義を設定するために、前記システムによって実行可能な命令を含むことを特徴とする請求項３９に記載の装置。

【請求項４３】　参加者インターフェースの定義を構築するための前記ツールは、

構築されているインターフェースの前記出力ドキュメントの定義のように、前記相補トランザクションの入力ドキュメントの定義を含むために、前記システムによって実行可能である命令を含むことを特徴とする請求項４２に記載の装置。

【請求項４４】　参加者インターフェースの定義を構築するための前記ツールは、特定のトランザクションの識別子を記録するための論理構造を含む参加者インターフェース・ドキュメントの定義に従ったドキュメントと、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つとを構築するために、前記システムによって実行可能である命令を含むことを特徴とする請求項３９に記載の装置。

【請求項４５】　参加者インターフェースの定義を構築するための前記ツー

ルは、前記参加者インターフェースの識別子を記録するため、及び前記参加者インターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含む参加者インターフェース・ドキュメントの定義に従ったドキュメントを構築するために、前記システムによって実行可能な命令を含むことを特徴とする請求項３９に記載の装置。

【請求項４６】　参加者インターフェース・ドキュメントの定義に従った前記ドキュメントは、特定のトランザクションの機械可読仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項４５に記載の装置。

【請求項４７】　前記記録装置は、パースされたデータを具備することを特徴とする請求項２５に記載の装置。

【請求項４８】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタをコード化するキャラクタ・データと；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識別するマーク付けデータとを具備することを特徴とする請求項４７に記載の装置。

【請求項４９】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項４８に記載の装置。

【請求項５０】　前記仕様は、前記入力及び出力ドキュメントのうち少なくとも一つの前記論理構造によって識別される前記記録装置の組のうち少なくとも一つに対する翻訳情報を含み、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項４７に記載の装置。

【請求項５１】　前記記録装置は、パースされていないデータを具備するこ

とを特徴とする請求項４７に記載の装置。

【請求項５２】　前記記録装置の組に対応するデータ構造及び前記入力及び出力ドキュメントの論理構造は、可変トランザクション処理アーキテクチャに従った変数及び方法を含むプログラミング・オブジェクトを含むことを特徴とする請求項３９に記載の装置。

【請求項５３】　前記トランザクション処理の前記可変トランザクション処理アーキテクチャは、インターフェース記述言語に従った処理を具備することを特徴とする請求項３９に記載の装置。

【請求項５４】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項３９に記載の装置。

【請求項５５】　前記入力及び出力ドキュメントの一つの前記定義は、レポジトリにおけるドキュメント・タイプへの参照を含むことを特徴とする請求項４１に記載の装置。

【請求項５６】　前記レポジトリは、トランザクションの製品サブジェクトの測定を特定する翻訳情報を含むことを特徴とする請求項４０に記載の装置。

【請求項５７】　前記レポジトリは、トランザクションの製品サブジェクトのコストを特定する翻訳情報を含むことを特徴とする請求項４０に記載の装置。

【請求項５８】　前記レポジトリは、トランザクションの製品サブジェクトの特性を特定する翻訳情報を含むことを特徴とする請求項４０に記載の装置。

【請求項５９】　前記レポジトリは、トランザクションの金融タームを特定する翻訳情報を含むことを特徴とする請求項４０に記載の装置。

【請求項６０】　前記レポジトリは、トランザクションの製品サブジェクトに対する出荷のタームを特定する翻訳情報を含むことを特徴とする請求項４０に記載の装置。

【請求項６１】　ネットワークにおいて商業トランザクションをプログラミングするための方法であって：

前記トランザクションにおける処理を実行するためのリソースを含むネットワークにおけるノードに対する入力ドキュメントの機械可読定義と、及び前記ノー

ドに対する出力ドキュメントの機械可読定義を定義する段階であって、前記入力及び出力ドキュメントの定義は、記録装置の組の個別の記述と及び前記記録装置の組に対する論理構造とを具備する前記段階と；及び

前記論理構造に対する翻訳情報を前記ノードに供給する段階とを具備することを特徴とする方法。

【請求項６２】　前記翻訳情報は、前記入力及び出力ドキュメントの定義における少なくとも一つの論理構造に対するデータ・タイプ仕様を含むことを特徴とする請求項６１に記載の方法。

【請求項６３】　前記翻訳情報は、前記入力及び出力ドキュメントの定義における特定の論理構造に対する予め決められた組の記録装置を、リストの個別のエントリにマッピングする少なくとも一つのデータ構造を含むことを特徴とする請求項前記翻訳情報は、前記入力及び出力ドキュメントの定義における少なくとも一つの論理構造に対するデータ・タイプの仕様を含むことを特徴とする請求項６１に記載の方法。

【請求項６４】　翻訳情報を供給する前記段階は、論理構造のライブラリと、及び論理構造に対する翻訳情報を記録する、ネットワークにおける少なくとも一つのノードによってアクセス可能であるメモリにおけるリポジトリを供給する段階を含むことを特徴とする請求項６１に記載の方法。

【請求項６５】　特定のトランザクションの識別子を記録するための論理構造と、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むインターフェース・ドキュメントの定義に従ったドキュメントを含むインターフェースの機械可読仕様を定義する段階を含むことを特徴とする請求項６１に記載の方法。

【請求項６６】　前記記録装置はパースされたデータを具備することを特徴とする請求項６１に記載の方法。

【請求項６７】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタをコード

化するキャラクタ・データと；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識別するマーク付けデータとを具備することを特徴とする請求項６６に記載の方法。

【請求項６８】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項６７に記載の方法。

【請求項６９】　前記入力及び出力ドキュメントの少なくとも一つの特定の論理構造によって識別される前記記録装置の組の少なくとも一つに対する前記翻訳情報は、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項６７に記載の方法。

【請求項７０】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項６６に記載の方法。

【請求項７１】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項６１に記載の方法。

【請求項７２】　前記入力ドキュメントの定義において、論理構造に応答してイベント信号（event signal）を生成するためのパーサ（parser）を供給する段階と；及び

前記処理を実行するために、前記イベント信号に応答するイベント・リスナ・プログラム（event listener programs）を供給する段階とを含むことを特徴とする請求項６１に記載の方法。

【請求項７３】　トランザクションに関連する処理を実行する複数のノードを含むネットワークにおけるノード間のトランザクションを実行するための方法であって：

トランザクションに対するインターフェースの機械可読仕様を記録する段階であって、前記仕様は入力ドキュメントの定義及び出力ドキュメントの定義を含み、前記入力及び出力ドキュメントの定義は、記録装置の組の個別の記述と及び前記記録装置の組に対する論理構造とを具備する前記段階と；

通信ネットワークを通してドキュメントを具備するデータを受信する段階と；

入力ドキュメントを識別するための前記仕様に従って前記ドキュメントをパースする段階と；

機械可読形式での前記入力ドキュメントの少なくとも一部分を、出力を生成するトランザクション処理に供給する段階と；

前記仕様に基づいて、出力ドキュメントの前記定義に従った前記出力を具備する出力ドキュメントを形成する段階と；及び

前記通信ネットワーク上で前記出力ドキュメントを転送する段階とを具備することを特徴とする方法。

【請求項７４】　前記トランザクションに対するネットワークにおける他のノードに供給される相補インターフェースの仕様にアクセスする段階であって、前記アクセスされた仕様は、前記相補インターフェースに対する入力ドキュメントの定義と、及び前記相補インターフェースに対する出力ドキュメントの定義とを含む前記段階と；及び

前記記録された仕様におけるインターフェースの前記入力ドキュメントの前記定義における前記相補インターフェースの前記出力ドキュメントの前記定義の少なくとも一部を含むことによって、前記インターフェースの前記記録された仕様を設定する段階とを含むことを特徴とする請求項７３に記載の方法。

【請求項７５】　ネットワークにおいて前記相補インターフェースを見つける段階を含むことを特徴とする請求項７４に記載の方法。

【請求項７６】　前記記録された仕様を設定する前記段階は、レポジトリからの機械可読仕様のエレメントにアクセスする段階であって、前記レポジトリは、論理構造のライブラリと、論理構造に対する模式図マップ（schematic map）と、及びインターフェース記述を構築するために使用される論理構造を具備するドキュメントの定義とを記録する前記段階を含むことを特徴とする請求項７４に記載の方法。

【請求項７７】　前記記録された仕様におけるインターフェースの前記出力ドキュメントの前記定義における前記相補インターフェースの前記入力ドキュメントの前記定義の少なくとも一部を含むことによって、前記インターフェースの

前記記録された仕様を設定する段階とを含むことを特徴とする請求項７４に記載の方法。

【請求項７８】　通信ネットワークを通した前記仕様へのアクセスを、ネットーワークにおける他のノードに供給する段階を含むことを特徴とする請求項７３に記載の方法。

【請求項７９】　前記インターフェースの仕様をネットワークにおける他のノードに送信する段階を含み、前記ネットワークにおいては、前記仕様へのアクセスはネットワークにおける他のノードに対して供給される前記段階を含むことを特徴とする請求項７３に記載の方法。

【請求項８０】　前記機械可読仕様は、特定のトランザクションの識別子を記録するための論理構造と、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項７３に記載の方法。

【請求項８１】　前記機械可読仕様は、インターフェースの識別子を記録するための、及び前記インターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項７３に記載の方法。

【請求項８２】　前記機械可読仕様は、特定のトランザクションの仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項８１に記載の方法。

【請求項８３】　前記記録装置はパースされた（parsed）データを具備することを特徴とする請求項７３に記載の方法。

【請求項８４】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタをコード

化するキャラクタ・データと；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識別するマーク付けデータとを具備することを特徴とする請求項８３に記載の方法。

【請求項８５】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項８４に記載の方法。

【請求項８６】　前記仕様は、前記入力及び出力ドキュメントのうち少なくとも一つの論理構造によって識別される前記記録装置の組のうち少なくとも一つに対する前記翻訳情報を含み、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項８３に記載の方法。

【請求項８７】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項８３に記載の方法。

【請求項８８】　前記トランザクション処理は、可変トランザクション処理アーキテクチャを有し、及び前記入力ドキュメントの少なくとも一部分を、前記トランザクション処理の前記可変トランザクション処理アーキテクチャに従って読み取り可能な形式に翻訳する段階を含むことを特徴とする請求項７３に記載の方法。

【請求項８９】　前記翻訳段階は、前記トランザクション処理の前記可変トランザクション処理アーキテクチャに従った変数及び方法を含むプログラミング・オブジェクトを生成する段階を含むことを特徴とする請求項８８に記載の方法。

【請求項９０】　前記トランザクション処理の前記可変トランザクション処理アーキテクチャは、インターフェース記述言語に従った処理を具備することを特徴とする請求項８８に記載の方法。

【請求項９１】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項７３に記載の方法。

【請求項９２】　複数のトランザクションにおける使用のためのドキュメン

ト・タイプのレポジトリを供給する段階を含み、及び前記入力及び出力ドキュメントの一つの前記定義は、前記レポジトリにおけるドキュメント・タイプへの参照を含むことを特徴とする請求項７３に記載の方法。

【請求項９３】　ドキュメント・タイプの前記レポジトリは、ネットワークにおける参加者処理を識別するためのドキュメント・タイプを含むことを特徴とする請求項９２に記載の方法。

【請求項９４】　論理構造に対する翻訳情報のレポジトリを供給する段階を含み、トランザクションのパラメータを識別する翻訳情報を含むことを特徴とする請求項９２に記載の方法。

【請求項９５】　前記トランザクション処理は、可変トランザクション処理アーキテクチャを有し、及び：

相補インターフェースの仕様にアクセスする段階であって、前記アクセスされた仕様は前記相補インターフェースに対する入力ドキュメントの定義と及び前記相補インターフェースに対する出力ドキュメントの定義とを含む前記段階と；

記録された仕様におけるインターフェースの前記入力ドキュメントの定義のように、前記相補インターフェースの前記出力ドキュメントの定義を含むことによって、インターフェースの記録された仕様を設定する段階と；及び

前記入力及び出力ドキュメントの定義に応答して、記録装置の組に対応したデータ構造及び前記トランザクション処理の前記トランザクション処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造と、前記入力ドキュメントを前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令と、及び前記トランザクション処理の出力を前記出力ドキュメントの定義に従った形式に翻訳するために、前記システムによって実行可能な命令とをコンパイルする段階とを含むことを特徴とする請求項７３に記載の方法。

【請求項９６】　記録された仕様におけるインターフェースの前記出力ドキュメントの定義のように、前記相補インターフェースの前記入力ドキュメントの定義を含むことによって、インターフェースの記録された仕様を設定する段階を含むことを特徴とする請求項９５に記載の方法。

【請求項９７】　前記トランザクション処理は、可変トランザクション処理

アーキテクチャを有し、及び前記記録された仕様を設定する段階は、レポジトリからの前記機械可読仕様のエレメントにアクセスする段階を含み、前記レポジトリは論理構造のライブラリと、論理構造に対する模式図マップと、及びインターフェース記述を構築するために使用される論理構造を具備するドキュメントの定義とを記録し、及び；

前記入力及び出力ドキュメントの定義に応答して、記録装置の組に対応したデータ構造及び前記トランザクション処理の前記トランザクション処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造と、前記入力ドキュメントを前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令と、及び前記トランザクション処理の出力を前記出力ドキュメントの定義に従った形式に翻訳するために、前記システムによって実行可能な命令とをコンパイルする段階とを含むことを特徴とする請求項７３に記載の方法。

【請求項９８】　トランザクションに関連する処理を実行する複数のノードを含むネットワークにおけるノード間のトランザクションを管理するための装置であって：

ネットワーク・インターフェースと；

命令のデータ及びプログラムを記録するメモリであって、トランザクションに対するインターフェースの機械可読仕様を含み、前記仕様は入力ドキュメントの定義と及び出力ドキュメントの定義とを含み、前記入力及び出力ドキュメントの定義は、記録装置の組の個別の記述と及び前記記録装置の組に対する論理構造とを具備する前記メモリと；

前記メモリと及び命令のプログラムを実行する前記ネットワーク・インターフェースとに接続されたデータ・プロセッサであって；前記命令のプログラムは、

ネットワーク・インターフェースを通してドキュメントを具備するデータを受信するための論理と；

入力ドキュメントを識別するための前記仕様に従って、前記ドキュメントをパースするための論理と；

機械可読形式での前記入力ドキュメントの少なくとも一部分を、出力を生成するトランザクション処理に供給するための論理と；

前記仕様に基づいて、出力ドキュメントの定義に従って、前記出力を具備する出力ドキュメントを形成するための論理と；及び

ネットワーク・インターフェース上で前記出力ドキュメントを転送するための論理とを含む前記データ・プロセッサとを具備することを特徴とする装置。

【請求項９９】　相補インターフェースの仕様にアクセスするための論理であって、前記アクセスされた仕様は前記相補インターフェースに対する入力ドキュメントの定義と及び前記相補インターフェースに対する出力ドキュメントの定義とを含む前記論理と；及び

記録された仕様における前記インターフェースの前記入力ドキュメントの定義のように、前記相補インターフェースの前記出力ドキュメントの定義を含むことによって、前記インターフェースの前記記録された仕様を設定するための論理とを含むことを特徴とする請求項９８に記載の装置。

【請求項１００】　前記記録された仕様を設定するための前記論理は、レポジトリからの前記機械可読仕様のエレメントにアクセスするための論理を含み、前記レポジトリは論理構造のライブラリと、論理構造に対する模式図マップと、及びインターフェース記述を構築するために使用される論理構造を具備するドキュメントの定義とを記録することを特徴とする請求項９９に記載の装置。

【請求項１０１】　記録された仕様における前記インターフェースの前記出力ドキュメントの定義のように、前記相補インターフェースの前記入力ドキュメントの定義を含むことによって、前記インターフェースの前記記録された仕様を設定するための論理を含むことを特徴とする請求項９９に記載の装置。

【請求項１０２】　通信ネットワークを通した前記仕様へのアクセスを、ネットーワークにおける他のノードに供給する論理を含むことを特徴とする請求項９８に記載の装置。

【請求項１０３】　前記インターフェースの仕様をネットワークにおける他のノードに送信するための論理を含み、前記ネットワークにおいては、前記仕様へのアクセスはネットワークにおける他のノードに対して供給される前記論理を含むことを特徴とする請求項９８に記載の装置。

【請求項１０４】　前記機械可読仕様は、特定のトランザクションの識別子を記録するための論理構造と、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項９８に記載の装置。

【請求項１０５】　前記機械可読仕様は、インターフェースの識別子を記録するための、及び前記インターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項９８に記載の装置。

【請求項１０６】　前記機械可読仕様は、特定のトランザクションの仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項１０５に記載の装置。

【請求項１０７】　前記記録装置はパースされたデータを具備することを特徴とする請求項９８に記載の装置。

【請求項１０８】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタをコード化するキャラクタ・データと；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識別するマーク付けデータとを具備することを特徴とする請求項１０７に記載の装置。

【請求項１０９】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項１０８に記載の装置。

【請求項１１０】　前記仕様は、前記入力及び出力ドキュメントのうち少なくとも一つの論理構造によって識別される前記記録装置の組のうち少なくとも一

データ構造及び前記トランザクション処理の前記トランザクション処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造とをコンパイルするためと、前記入力ドキュメントを前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令をコンパイルするためと、及び前記トランザクション処理の出力を前記出力ドキュメントの定義に従った形式に翻訳するために、前記システムによって実行可能な命令をコンパイルする論理を含むことを特徴とする請求項９８に記載の装置。

【請求項１２２】　トランザクションに関連する処理を実行する複数のノードを含むネットワークにおけるノード間のトランザクションを管理するための装置であって：

複数の参加者インターフェースの機械可読仕様を記録する段階であって、前記参加者インターフェースはトランザクションを識別し、前記個別のトランザクションは入力ドキュメントの定義及び出力ドキュメントの定義によって識別され、前記入力及び出力ドキュメントの前記定義は、記録装置の組の個別の記述と及び前記記録装置の組に対する論理構造とを具備する前記段階と；

通信ネットワークを通してドキュメントを具備するデータを受信する段階と；

入力ドキュメントを識別するための前記仕様と、及び前記識別された入力ドキュメントを受信する一つ以上のトランザクションに従って、前記ドキュメントをパースする段階と；

機械可読形式での前記入力ドキュメントの少なくとも一部分を、前記一つ以上の識別されたトランザクションと関連したトランザクション処理に供給する段階とを具備することを特徴とする方法。

【請求項１２３】　論理構造のライブラリと、論理構造に対する模式図マップと、及び参加者インターフェース記述を構築するために使用される論理構造を具備するドキュメントの定義とを記録するレポジトリを供給する段階を含むことを特徴とする請求項１２２に記載の方法。

【請求項１２４】　通信ネットワークを通した前記レポジトリへのアクセスを、ネットーワークにおける他のノードに供給する段階を含むことを特徴とする請求項１２３に記載の方法。

【請求項１２５】　前記機械可読仕様は、特定のトランザクションの識別子を記録するための論理構造と、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つとを含む参加者インターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項１２２に記載の方法。

【請求項１２６】　前記機械可読仕様は、参加者インターフェースの識別子を記録するための、及び前記参加者インターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項１２２に記載の方法。

【請求項１２７】　参加者インターフェース・ドキュメントの定義に従った前記ドキュメントは、特定のトランザクションの仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項１２６に記載の方法。

【請求項１２８】　前記記録装置はパースされたデータを具備することを特徴とする請求項１２２に記載の方法。

【請求項１２９】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタをコード化するキャラクタ・データと；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識別するマーク付けデータとを具備することを特徴とする請求項１２８に記載の方法。

【請求項１３０】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項１０９に記載の方法。

【請求項１３１】　前記仕様は、前記入力及び出力ドキュメントのうち少な

くとも一つの論理構造によって識別される前記記録装置の組のうち少なくとも一つに対する前記翻訳情報を含み、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項１３０に記載の方法。

【請求項１３２】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項１３０に記載の方法。

【請求項１３３】　機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給する段階は、処理アーキテクチャに従ってルーティング処理を実行する段階を含み、及び：

前記参加者インターフェースにおける前記入力及び出力ドキュメントの定義に応答して、記録装置の前記組に対応するデータ構造及び前記トランザクション処理の処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造と、及び前記入力ドキュメントを前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令とをコンパイルする段階を含むことを特徴とする請求項１２２に記載の方法。

【請求項１３４】　機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給する段階は、処理アーキテクチャに従ってルーティング処理を実行する段階と、及び前記入力ドキュメントの少なくとも一部分を、前記処理アーキテクチャに従って読み取り可能である形式に翻訳する段階を含むことを特徴とする請求項１２２に記載の方法。

【請求項１３５】　前記翻訳段階は、前記ルーティング処理の前記処理アーキテクチャに従った変数及び方法を含むプログラミング・オブジェクトを生成する段階を含むことを特徴とする請求項１３４に記載の方法。

【請求項１３６】　機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給する段階は、前記入力ドキュメントの前記部分を前記識別されたトランザクションにルーティングする段階を含むことを特徴とする請求項１２２に記載の方法。

【請求項１３７】　前記ルーティング段階は、通信ネットワーク上の前記入力ドキュメントを、前記識別されたトランザクションの一つを実行するノードに送信する段階を含むことを特徴とする請求項１３６に記載の方法。

【請求項１３８】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項１２２に記載の方法。

【請求項１３９】　参加者インターフェースの仕様は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義に従ってドキュメントの定義を具備することを特徴とする請求項１３８に記載の方法。

【請求項１４０】　前記レポジトリは複数のトランザクションにおける使用のための標準化されたドキュメント・タイプを含み、及び前記入力及び出力ドキュメントの一つの前記定義は、レポジトリにおける標準化されたドキュメント・タイプへの参照を含むことを特徴とする請求項１２２に記載の方法。

【請求項１４１】　前記レポジトリは、ネットワークにおける参加者処理を識別するための標準化されたドキュメント・タイプを含むことを特徴とする請求項１４０に記載の方法。

【請求項１４２】　論理構造に対する翻訳情報のレポジトリを供給する段階を含み、トランザクションのパラメータを識別する翻訳情報を含むことを特徴とする請求項１４０に記載の方法。

【請求項１４３】　前記トランザクション処理は、複数の可変トランザクション処理アーキテクチャの一つを各々有し、及び前記入力ドキュメントの少なくとも一部分を、前記個別のトランザクション処理の前記可変トランザクション処理アーキテクチャに従って読み取り可能な形式に翻訳する段階と、及び前記翻訳された部分を前記個別のトランザクション処理にルーティングする段階とを含むことを特徴とする請求項１２２に記載の方法。

【請求項１４４】　前記翻訳段階は、前記個別のトランザクション処理の可変トランザクション処理アーキテクチャに従って変数及び方法を含むプログラミング・オブジェクトを生成する段階を含むことを特徴とする請求項１４３に記載の方法。

【請求項１４５】　前記トランザクション処理の前記可変トランザクション処理アーキテクチャは、インターフェース記述言語に従った処理を具備することを特徴とする請求項１４４に記載の方法。

【請求項１４６】　トランザクションに関連する処理を実行する複数のノードを含むネットワークにおけるノード間のトランザクションを管理するための装置であって：

ネットワーク・インターフェースと；

命令のデータ及びプログラムを記録するメモリであって、複数の参加者インターフェースの機械可読仕様を含み、前記参加者インターフェースはトランザクションを識別し、前記個別のトランザクションは、入力ドキュメントの定義及び出力ドキュメントの定義によって識別され、前記入力及び出力ドキュメントの定義は、記録装置の組の個別の記述と及び前記記録装置の組に対する論理構造とを具備する前記メモリと；

前記メモリと及び命令のプログラムを実行する前記ネットワーク・インターフェースとに接続されたデータ・プロセッサであって；前記命令のプログラムは、

ネットワーク・インターフェースを通してドキュメントを具備するデータを受信するための論理と；

入力ドキュメントを識別するための仕様及び識別された入力ドキュメントを受信する一つ以上のトランザクションに従って、前記ドキュメントをパースするための論理と；及び

機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給するための論理とを含む前記データ・プロセッサとを具備することを特徴とする装置。

【請求項１４７】　論理構造のライブラリと、論理構造に対する模式図マップと及び参加者インターフェース記述を構築するために使用される論理構造を具備するドキュメントの定義を記録する前記データ・プロセッサによってアクセス可能なメモリに記録されたレポジトリを含むことを特徴とする請求項１４６に記載の装置。

【請求項１４８】　論理構造のライブラリと、論理構造に対する模式図マップと及び参加者インターフェース記述を構築するために使用される論理構造を具備するドキュメントの定義を記録するネットワーク・インターフェースを通してメモリに記録されるレポジトリへアクセスするための論理を含むことを特徴とする請求項１４６に記載の装置。

【請求項１４９】　前記機械可読仕様は、特定のトランザクションの識別子を記録するための論理構造と、及び前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つとを含む参加者インターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項１４６に記載の装置。

【請求項１５０】　前記機械可読仕様は、参加者インターフェースの識別子を記録するための、及び前記参加者インターフェースによってサポートされる一組の一つ以上のトランザクションの仕様及び仕様への参照のうち少なくとも一つを記録するための論理構造を含むインターフェース・ドキュメントの定義に従ったドキュメントを含むことを特徴とする請求項１４６に記載の装置。

【請求項１５１】　参加者インターフェース・ドキュメントの定義に従った前記ドキュメントは、特定のトランザクションの仕様への参照を含み、及び前記特定のトランザクションの仕様は、前記特定のトランザクションに対する入力及び出力ドキュメントの定義及び定義への参照のうち少なくとも一つを記録するための論理構造を含むドキュメントを含むことを特徴とする請求項１５０に記載の装置。

【請求項１５２】　前記記録装置はパースされたデータを具備することを特徴とする請求項１４６に記載の装置。

【請求項１５３】　前記入力及び出力ドキュメントの少なくとも一つにおける前記パースされたデータは：

前記入力及び出力ドキュメントの一つにおけるテキスト・キャラクタをコード化するキャラクタ・データと；及び

記録装置の組を、前記入力及び出力ドキュメントの一つの論理構造に従って識別するマーク付けデータとを具備することを特徴とする請求項１５２に記載の装

置。

【請求項１５４】　前記記録装置の組の少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化することを特徴とする請求項１５３に記載の装置。

【請求項１５５】　前記仕様は、前記入力及び出力ドキュメントのうち少なくとも一つの論理構造によって識別される前記記録装置の組のうち少なくとも一つに対する前記翻訳情報を含み、パースされたキャラクタの組に対する個別の定義をコード化することを特徴とする請求項１５４に記載の装置。

【請求項１５６】　前記記録装置は、パースされていないデータを具備することを特徴とする請求項１５４に記載の装置。

【請求項１５７】　機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給するための論理は、処理アーキテクチャに従ったルーティング処理を含み、及び：

前記参加者インターフェースにおける前記入力及び出力ドキュメントの定義に応答して、記録装置の前記組に対応するデータ構造及び前記トランザクション処理の処理アーキテクチャに従った前記入力及び出力ドキュメントの論理構造とをコンパイルするためと、及び前記入力ドキュメントを前記対応するデータ構造に翻訳するために、前記システムによって実行可能な命令とをコンパイルするためのコンパイラを含むことを特徴とする請求項１４６に記載の装置。

【請求項１５８】　機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給するための論理は、処理アーキテクチャに従ったルーティング処理を含み、及び：前記入力ドキュメントの少なくとも一部分を、前記処理アーキテクチャに従って読み取り可能である形式に翻訳するための論理を含むことを特徴とする請求項１４６に記載の装置。

【請求項１５９】　前記翻訳するための論理は、前記ルーティング処理の処理アーキテクチャに従った変数及び方法を含むプログラミング・オブジェクトを生成する段階を含むことを特徴とする請求項１５８に記載の装置。

【請求項１６０】　機械可読形式での前記入力ドキュメントの少なくとも一部分を、一つ以上の識別されたトランザクションと関連するトランザクション処理に供給するための論理は、前記入力ドキュメントの前記部分を前記識別されたトランザクションにルーティングするためのルータ（router）を含むことを特徴とする請求項１４６に記載の装置。

【請求項１６１】　前記ルータは、ネットワーク・インターフェース上の前記入力ドキュメントを、前記識別されたトランザクションの一つを実行するノードに送信するための論理を含むことを特徴とする請求項１６０に記載の装置。

【請求項１６２】　前記入力及び出力ドキュメントの定義は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義を具備することを特徴とする請求項１４６に記載の装置。

【請求項１６３】　参加者インターフェースの仕様は、標準拡張可能マーク付け言語ＸＭＬに従ったドキュメント・タイプ定義に従ってドキュメントの定義を具備することを特徴とする請求項１６２に記載の装置。

【請求項１６４】　前記レポジトリは複数のトランザクションにおける使用のための標準化されたドキュメント・タイプを含み、及び前記入力及び出力ドキュメントの一つの前記定義は、レポジトリにおける標準化されたドキュメント・タイプへの参照を含むことを特徴とする請求項１４７に記載の装置。

【請求項１６５】　前記レポジトリは、ネットワークにおける参加者処理を識別するための標準化されたドキュメント・タイプを含むことを特徴とする請求項１４７に記載の装置。

【請求項１６６】　前記トランザクション処理は、複数の可変トランザクション処理アーキテクチャの一つを各々有し、及び前記入力ドキュメントの少なくとも一部分を、前記個別のトランザクション処理の前記可変トランザクション処理アーキテクチャに従って読み取り可能な形式に翻訳するため、及び前記翻訳された部分を前記個別のトランザクション処理にルーティングするための論理を含むことを特徴とする請求項１４６に記載の装置。

【請求項１６７】　前記翻訳するための論理は、前記個別のトランザクション処理の前記可変トランザクション処理アーキテクチャに従って変数及び方法を

含むプログラミング・オブジェクトを生成することを特徴とする請求項１６６に記載の装置。

【請求項１６８】　前記トランザクション処理の前記可変トランザクション処理アーキテクチャは、インターフェース記述言語に従った処理を具備することを特徴とする請求項１６６に記載の装置。

【請求項１６９】　ネットワークのトレーディングパートナ間のトランザクションを確立するための方法であって、

トレーディングパートナにより提供されたビジネスサービスを明記する機械可読仕様であって、提供されたサービスの定義及びその定義への参照のうち少なくとも一つ、及びトレーディングパートナによるそのようなサービスと交換されるドキュメントの定義及びその定義への参照のうちの少なくとも一つを含む機械可読仕様のレジストリを維持し、

リクエストに応じて、前記レジストリからの１以上の機械可読仕様を通信ネットワークを介してリクエストノードに供給する、

ことを含むことを特徴とする方法。

【請求項１７０】　前記機械可読仕様は、記憶装置セットのそれぞれの記述及び該記憶装置セットの論理構造を識別するデータを備えている請求項１６９に記載の方法。

【請求項１７１】　前記機械可読仕様は、入力ドキュメント及び該入力ドキュメントを受け入れる１以上のトランザクションを識別するためパースするようになっているデータを含んでいる請求項１７０に記載の方法。

【請求項１７２】　交換される前記ドキュメントの定義は、記憶装置セットのそれぞれの記述と、該記憶装置セットの論理構造とを含んでいる請求項１６９に記載の方法。

【請求項１７３】　前記機械可読仕様は、特定のトランザクションの識別子を記憶する論理構造を含む所定のドキュメントの定義に従順なドキュメントと、特定のトランザクションのための入出力ドキュメントの定義及びその定義への参照のうち少なくとも一つとを含んでいる請求項１６９に記載の方法。

【請求項１７４】　前記記憶装置はパースされたデータを含んでいる請求項

１７０に記載の方法。

【請求項１７５】　交換された少なくとも一つのドキュメントの前記パースされたデータは、

前記入出力ドキュメントの一つのテキスト・キャラクタをコード化するキャラクタ・データと、

前記入出力ドキュメントの一つの論理構造により記憶装置セットを識別するマークアップデータと、

を備えている請求項１７４に記載の方法。

【請求項１７６】　前記記憶装置セットの少なくとも一つは、自然言語の単語を供給する複数のテキスト・キャラクタをコード化する請求項１７５に記載の方法。

【請求項１７７】　前記仕様は、前記入出力ドキュメントのうち少なくとも一つの論理構造により識別された記憶装置セットの少なくとも一つのための翻訳情報を含み、パースされたキャラクタセットのためのそれぞれの定義をコード化する請求項１７５に記載の方法。

【請求項１７８】　前記記憶装置は、パースされないデータを含んでいる請求項１７４に記載の方法。

【請求項１７９】　前記機械可読仕様と関連するトレーディングパートナを含んでいる請求項１６９に記載の方法。